Development of FPGA based Standalone Tunable Fuzzy Logic Controllers

Bhaskara Rao Jammu

Department of Electronics and Communications Engineering
National Institute of Technology Rourkela

Development of FPGA based Standalone Tunable Fuzzy Logic Controllers

Dissertation submitted in partial fulfillment

of the requirements of the degree of

Doctor of Philosophy

in

Electronics and Communications Engineering

by

Bhaskara Rao Jammu

(Roll Number: 511EC103)

based on research carried out

under the supervision of

Prof. Sarat Kumar Patra

and

Prof. Kamala Kanta Mahapatra

July, 2017

Department of Electronics and Communications Engineering

National Institute of Technology Rourkela

Department of Electronics and Communications Engineering
National Institute of Technology Rourkela

July 17, 2017

Certificate of Examination

Roll Number: *511EC103*

Name: *Bhaskara Rao Jammu*

Title of Dissertation: *Development of FPGA based Standalone Tunable Fuzzy Logic Controllers*

We the below signed, after checking the dissertation mentioned above and the official record book (s) of the student, hereby state our approval of the dissertation submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Department of Electronics and Communications Engineering at National Institute of Technology Rourkela. We are satisfied with the volume, quality, correctness, and originality of the work.

Kamala Kanta Mahapatra Co-Supervisor	Sarat Kumar Patra Principal Supervisor
Bidyadhar Subudhi Member, DSC	Debiprasad Priyabrata Acharya Member, DSC
Dayal Ramakrushna Parhi Member, DSC	Indra Narayan Kar External Examiner
Sukadev Meher Chairperson, DSC	Kamala Kanta MahaPatra Head of the Department

Department of Electronics and Communications Engineering
National Institute of Technology Rourkela

Prof. Sarat Kumar Patra
Professor

Prof. Kamala Kanta Mahapatra
Professor

July 17, 2017

Supervisors' Certificate

This is to certify that the work presented in the dissertation entitled *Development of FPGA based Standalone Tunable Fuzzy Logic Controllers* submitted by *Bhaskara Rao Jammu*, Roll Number 511EC103, is a record of original research carried out by him under our supervision and guidance in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Department of Electronics and Communications Engineering*. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Kamala Kanta Mahapatra Professor	Sarat Kumar Patra Professor

Dedication

I dedicate my thesis to My Mother and My Child Taruni...

Signature

Declaration of Originality

I, *Bhaskara Rao Jammu*, Roll Number *511EC103* hereby declare that this dissertation entitled *Development of FPGA based Standalone Tunable Fuzzy Logic Controllers* presents my original work carried out as a doctoral student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections "Reference" or "Bibliography". I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

July 17, 2017
NIT Rourkela

Bhaskara Rao Jammu

Acknowledgment

I owe sincere gratitude to the ones who have contributed greatly to completion of this thesis.

First and foremost I would like to express my sincerest appreciation to my supervisor, Prof. Sarat Kumar Patra, who has guided me throughout my Ph.D. thesis with his patience and knowledge whilst allowing me to work in my own way. It was an honor for me to work with him during my time at NIT Rourkela. I would also like to acknowledge my co-supervisor, Prof. Kamala Kanta Mahapatra for his kind advice and inspiration. "*Agnyaana Timiraandhasya Gnyaana Anjana Shalaakayaa Chakshuhu Unmeelitam Yenam Tasmai Sri Gurave Namaha* ".

I would like to thank Board of Research for Fusion Science and Technology (BRFST) and Institute of Plasma Research, Gandhinagar for funding a major part of this research. The author would like to extend his gratitude towards Dr. Govindarajan, Mr.J. J. Patel, Mrs. Rachana Rajpal and Mr. Hitesh Patel of Institute of Plasma Research, Gandhinagar, for their contributions to this project.

I am thankful to all the faculty members of Electronics and Communication Engineering department for extending their valuable suggestions and help whenever I approached.

I would like to thank my friends Pallab, Bijay, Manas, Satyendra, Chitra, Varun, Mangal, Govind, Rama Krishna, Tom, Srinivas, Sujeevan and others who were with me in the ups and downs of my life during my Ph.D. work. I would like to extend gratitude to my seniors Prasant, Karuppanan, Venkat, Rajesh, Kanhu, Trilochan, Yogesh, Manab and Dipak.

I would also like to thank Prof. B. Subudhi, Prof. D. P. Acharya, Prof. D. R. Parhi and Prof. S. Meher for their innovative ideas and review during the entire duration of the project.

I do acknowledge the academic resources that I have received from NIT Rourkela. I also thank the administrative and technical staff members of Electronics and Communication Engineering Department for their in time support.

In addition, I thank Dr. K V L Raju, Dr. R. Ramana Reddy and colleagues of MVGR College of Engineering for providing good workplace and encouragement. I also thank Dr.

M. V. S. Sairam, Dr. N. Bala Subramanyam, Dr. N. Balaji, the faculty of GVP College of Engineering and friends at JNTU Vizianagaram for their constant love and encouragement.

I take this opportunity to express my regards and obligation to my father whose support and encouragement I can never forget in my life. I feel proud to acknowledge my father for his throughout support and motivation in my career for whom I am today and always. I would like to dedicate this thesis to my wife Nalini and my daughter Nikhila for their unconditional love, patience, and cooperation.

Finally, there is no word to describe my gratitude toward my other family members for their endless support and love during my life.

Jul 17, 2017 *Bhaskara Rao Jammu*

NIT Rourkela Roll Number: 511EC103

Abstract

Soft computing techniques differ from conventional (hard) computing, in that unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind and its ability to address day-to-day problems. The principal constituents of Soft Computing (SC) are Fuzzy Logic (FL), Evolutionary Computation (EC), Machine Learning (ML) and Artificial Neural Networks (ANNs).

This thesis presents a generic hardware architecture for type-I and type-II standalone tunable Fuzzy Logic Controllers (FLCs) in Field Programmable Gate Array (FPGA). The designed FLC system can be remotely configured or tuned according to expert operated knowledge and deployed in different applications to replace traditional Proportional Integral Derivative (PID) controllers. This re-configurability is added as a feature to existing FLCs in literature. The FLC parameters which are needed for tuning purpose are mainly input range, output range, number of inputs, number of outputs, the parameters of the membership functions like slope and center points, and an If-Else rule base for the fuzzy inference process. Online tuning enables users to change these FLC parameters in real-time and eliminate repeated hardware programming whenever there is a need to change. Realization of these systems in real-time is difficult as the computational complexity increases exponentially with an increase in the number of inputs. Hence, the challenge lies in reducing the rule base significantly such that the inference time and the throughput time is perceivable for real-time applications.

To achieve these objectives, Modified Rule Active 2 Overlap Membership Function (MRA2-OMF), Modified Rule Active 3 Overlap Membership Function (MRA3-OMF), Modified Rule Active 4 Overlap Membership Function (MRA4-OMF), and Genetic Algorithm (GA) base rule optimization methods are proposed and implemented. These methods reduce the effective rules without compromising system accuracy and improve the cycle time in terms of Fuzzy Logic Inferences Per Second (FLIPS). In the proposed system

architecture, the FLC is segmented into three independent modules, fuzzifier, inference engine with rule base, and defuzzifier.

Fuzzy systems employ fuzzifier to convert the real world crisp input into the fuzzy output. In type 2 fuzzy systems there are two fuzzifications happen simultaneously from upper and lower membership functions (UMF and LMF) with subtractions and divisions. Non-restoring, very high radix, and newton raphson approximation are most widely used division algorithms in hardware implementations. However, these prevalent methods have a cost of more latency. In order to overcome this problem, a successive approximation division algorithm based type 2 fuzzifier is introduced. It has been observed that successive approximation based fuzzifier computation is faster than the other type 2 fuzzifier.

A hardware-software co-design is established on Virtex 5 LX110T FPGA board. The MATLAB Graphical User Interface (GUI) acquires the fuzzy (type 1 or type 2) parameters from users and a Universal Asynchronous Receiver/Transmitter (UART) is dedicated to data communication between the hardware and the fuzzy toolbox. This GUI is provided to initiate control, input, rule transfer, and then to observe the crisp output on the computer. A proposed method which can support canonical fuzzy IF-THEN rules, which includes special cases of the fuzzy rule base is included in Digital Fuzzy Logic Controller (DFLC) architecture. For this purpose, a mealy state machine is incorporated into the design. The proposed FLCs are implemented on Xilinx Virtex-5 LX110T. DFLC peripheral integration with Micro-Blaze (MB) processor through Processor Logic Bus (PLB) is established for Intellectual Property (IP) core validation. The performance of the proposed systems are compared to Fuzzy Toolbox of MATLAB. Analysis of these designs is carried out by using Hardware-In-Loop (HIL) test to control various plant models in MATLAB/Simulink environments.

***Keywords*: *Fuzzy Logic Controller*$_1$; *FPGA*$_2$; *GA*$_3$; *UART*$_4$; *HIL*$_5$.**

Contents

List of Figures

List of Tables

List of Notations

μ	Membership function
μ_A	Memebership function of fuzzy set A
$\bigcap$	T-norm operator. Basic operation includes Minimum, Product, Lukasiewicz, etc. Operated on a vector
$\bigcup$	T-conorm operator. Basic operation includes Maximum, Product, Lukasiewicz, etc
$\tilde{A}$	Type 2 Fuzzy Set of A
$\bar{A}$	Upper membership function of $\tilde{A}$
$\tilde{A}'$	Compliment of Type 2 fuzzy set $\tilde{A}$
N_{cells}	Total rule dimension
F_A	Fraction of MRA2-OMF rules with 2-OMF rules
F_{RA}	Fraction of MRA2-OMF rules with Total rules
F_{3A}	Fraction of MRA3-OMF rules with 3-OMF rules
F_{3RA}	Fraction of MRA3-OMF rules with Total rules
F_{4A}	Fraction of MRA4-OMF rules with 4-OMF rules
F_{4RA}	Fraction of MRA4-OMF rules with Total rules
D_k	k^{th} Data point in the data set
C_i	i^{th} Cluster
$d_{k,i}$	Distance of D_k from R_i, i.e the cluster center C_i
t_0	Start time of chaotic time series
Γ	Shafranov parameter
μ_o	Permeability of Vacuum
β_p	Poloidal beta
I_p	Plasma current
I_c	Coil and conductor current
V_c	Control voltage
B_v	Vertical magnetic field
l_i	Internal inductance of plasma magnetic field
Δt	Time interval of chaotic time series

List of Acronyms

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Integrated Processor
CMOS	Complementary Metal Oxide Semiconductor
COA	Centroid of Area
COG	Center of Gravity
DAC	Digital to Analog Converter
DFLC	Digital Fuzzy Logic Controller
DSP	Digital Signal Processor
EDA	Electronic Design Automation
EDK	Embedded Development Kit
FIE	Fuzzy Inference Engine
FF	Flip Flop
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FLT	Fuzzy Logic Toolbox
FLCS	Fuzzy Logic Control System
FLIPS	Fuzzy Logic Inferences Per Second
FPAA	Field Programmable Analog Array
FPGA	Field Programmable Gate Array
FOU	Footprint Of Uncertainty
FSM	Finite State Machine
GA	Genetic Algorithm
GA-FLC	Genetic Algorithm based Fuzzy Logic Controller
GUI	Graphical User Interface
GUIDE	Graphical User Interface Design Environment
HDL	Hardware Description Language
HIL	Hardware In Loop
IC	Integrated Circuit
IP	Intellectual Property
ISE	Integrated Software Environment
IT2FLC	Iterative Type 2 Fuzzy Logic Controller
KM	Karnik and Mendel
LMF	Lower Membership Function
LUT	Look Up Table
MB	Micro Blaze
MF	Membership Function

MFLIPS	Mega Fuzzy Logic Inferences Per Second
MFG	Membership Function Generator
MIMO	Multi Input Multi Output
MRA2-OMF	Modified Rule Active 2 Overlap Membership Function
MRA3-OMF	Modified Rule Active 3 Overlap Membership Function
MRA4-OMF	Modified Rule Active 4 Overlap Membership Function
OMF	Overlapping Membership Function
PAMA	Programmable Analog Multiplex Array
PC	Personal Computer
PID	Proportional Integral Derivative
PCI	Peripheral Component Interconnect
PLB	Programmable Logic Bus
PWM	Pulse Width Modulator
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register Transfer Level
SAIT2FLC	Successive Approximation based Iterative Type 2 Fuzzy Logic Controller
SDK	Software Development Kit
SFS	Single Fuzzy Statement
SoC	System on Chip
SM	System Method
T1FLC	Type 1 Fuzzy Logic Controller
T2FLC	Type 2 Fuzzy Logic Controller
TB	Test Bench
TFTR	Tokamak Fusion Test Reactor
VLSI	Very Large Scale Integration
UART	Universal Asynchronous Receiver and Transmitter
UMF	Upper Membership Function
WM	Wu-Mendel
WS	Work Station

Chapter 1

Background and Related Work

Preface

This chapter presents a brief discussion on some of the earlier works related to hardware implementations of fuzzy systems. The fuzzy system, its working principle, and the fundamental concepts are discussed. This chapter also addresses the issues of re-configurability and generality of the existing fuzzy system designs. The limitations of the current systems lead to the motivation for new work. The limitations along with the challenges and research areas are depicted in this chapter. Finally, the workflow of the present dissertation is summarized.

"There are things known and there are things unknown, and in between are the doors of perception."

Aldous Huxley

Global technologies evolution triggered increasing complexity of applications leading to new developments both in the industry and in the scientific research fields. Fuzzy control methods represent rather a different approach to the problems of controlling these complex nonlinear systems. Fuzzy logic and the theory of fuzzy sets are the results of a broader comprehension of practical control problems and control actions performed by human operators, which could not have been correctly interpreted by using classical bivalent logic and conventional methods of automatic control. Fuzzy logic is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. These can be implemented in hardware, software, or a combination of both. Fuzzy Logic Controller (FLC) provides an easy way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. The approach to control problems mimics how a human being would make decisions. However, requiring precision in engineering problems incurs a high cost and long time in development. Lotfi Askar Zadeh [2] described the power of uncertainty and approximate reasoning over hard computing by illustrating how a human mind works while *parking a vehicle*. T. Ross took the instance of *traveling salesman* problem to exemplify similar point [3]. It is, therefore, possible for scientist or engineer to contemplate the requirement for approximate reasoning and imprecision while considering fuzzy logic to solve a problem. The prime desideratum is "how much imprecision can the system tolerates".

1.1 Fuzzy Logic Systems - An Overview

As a general principle, a good engineering theory should be capable of making use of all available information effectively. For many practical systems, valuable information comes from two sources: one source is human experts who describe their knowledge about the

system in natural languages; the other is sensory measurements and mathematical models that are derived according to physical laws. An important task, therefore, is to combine these two types of information into system designs. To achieve this combination, a key question is how to formulate human knowledge into a similar framework which will be used to formulate sensory measurements and mathematical models. In other words, the fundamental issue is how to transform a human knowledge base into a mathematical formula. Essentially, a fuzzy system performs this transformation. To understand how this transformation is done, fuzzy systems are studied. Fuzzy systems are knowledge-based or rule-based systems. The heart of a fuzzy system is a knowledge base consisting of the so-called fuzzy IF-THEN rules. A fuzzy IF-THEN rule is an IF-THEN statement in which some words are characterized by continuous membership functions. Lotfi Askar Zadeh [4] defined these fuzzy sets and membership functions as **a class of objects with a continuum of grades of membership. Such a set is characterized by a Membership Function (MF) that assigns to each object, a grade of membership ranging from zero to one**. Fuzzy logic is useful to the people who are involved in research and development which includes engineers, mathematicians, medical researchers, business analysts, and natural scientists. Indeed, the applications of fuzzy logic can be found in many engineering and scientific works like washing machines, vacuum cleaners, antiskid braking systems, unmanned automobiles, weather forecasting systems, transmission systems, medical diagnosis and treatment plans, stock trading, etc.

1.1.1 Fuzzy Sets

There is an inherent impreciseness present in our natural language when we describe phenomena that do not have sharply defined boundaries. Statements such as "Tom is smart" and "Lorenzo is young" are simple examples. Fuzzy sets are mathematical objects modeling this impreciseness. The fuzzy set theory provides mathematical tools for carrying out approximate reasoning processes when available information is uncertain, incomplete, imprecise, or vague. Conventional bivalent set theory, often known as a conventional set theory, can be limiting in describing a 'humanistic' problem mathematically. For example, Figure 1.1 illustrates bivalent sets to model room temperature.

The limiting feature of conventional sets is that they are mutually exclusive, and it is impossible to have a membership of more than one set. Based on human perception, it is an inaccurate model to define a transition from quantity '**cool**' to '**warm**' when one degree

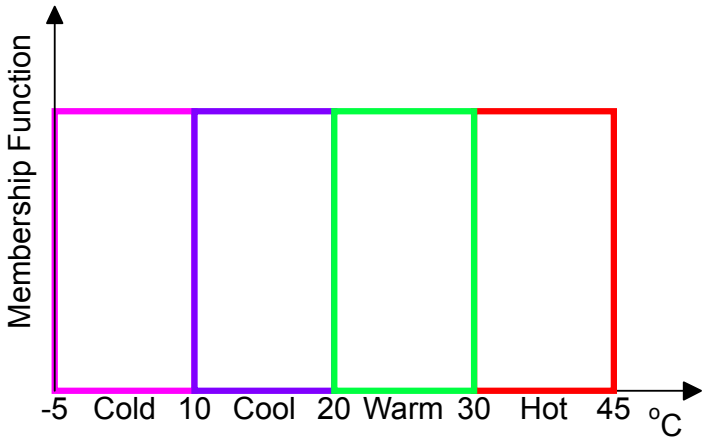


Figure 1.1: Conventional sets to model the room temperature.

centigrade of heat is added to the system. In the real world, the actual modeling occurs with a smooth drift or transition from '**cool**' to '**warm**'. This transition can be captured more accurately by Fuzzy Set Theory. Figure 1.2 shows fuzzy sets quantifying the same information which better describes this natural drift. Here, the association is modeled as a triangular function. In fuzzy logic theory, the function which defines the association is called as a membership function. Thereby, in fuzzy set theory, apart from the value of the variable, the degree of association of the variable to the set is also captured.

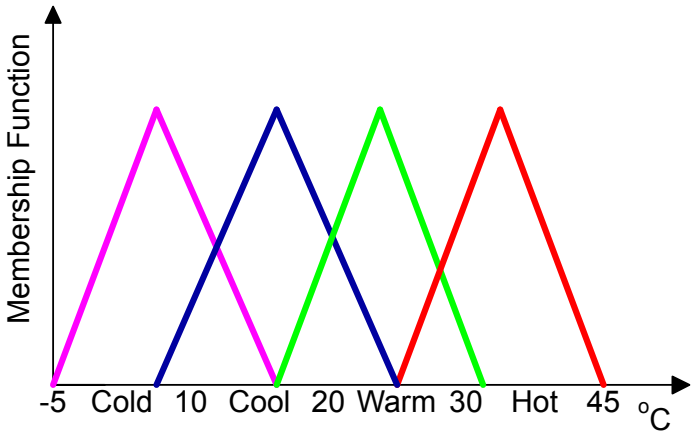


Figure 1.2: Fuzzy sets to model the room temperature.

Mathematically, U be the universe of discourse, or universal set, which contains all the possible elements of concern in each particular context or application. A fuzzy set A in U may be represented as a set of ordered pairs of a generic element x and its membership value, that is,

$$A = (x, \mu_A(x))|x \in U \tag{1.1}$$

Table 1.1: T-norm duals in fuzzy literature

t-norm	t-conorm	Description
Min(μ_y,μ_x)	Max(μ_y,μ_x)	Min/Max
$\mu_y\mu_x$	μ_y+ μ_x - $\mu_y\mu_x$	Product/Probabilistic Sum
Max(0,μ_y+μ_x-1)	Min(1,μ_y+ μ_x)	Bold Union/Bounded Sum

where μ_A(x) is called "membership function" (or MF for short) for the fuzzy set A [4]. The MF maps each element of X to a membership grade (or membership value) between 0 and 1.

A set is called support if $\{x|\mu_A(x) > 0\}$ and core if $\{x|\mu_A(x) = 1\}$. The set can be termed as normal if the core is nonempty, and fuzzy singleton if the support is single point in U if $\mu_A(x)$=1 [3].

Fuzzy mathematics provides the starting point and basic language for fuzzy systems and fuzzy control. A fuzzy system operates on various fuzzy sets to provide a suitable output. It is often required that these fuzzy sets are combined meaningfully. It is imperative that there exists a commonality of operators between regular and fuzzy sets. These operators are termed as *aggregators* [5].

1.1.2 Fuzzy Set Operations

Corresponding to ordinary set operations of the union (OR), intersection (AND) and complement (NOT), fuzzy sets have similar operations, which were initially defined in Zadeh's seminal work [4]. The Zadeh defines these operations by considering μ_x and μ_y as membership grade of two fuzzy numbers x and y, in a fuzzy set:

$$T(\mu_x, \mu_y) = \min(\mu_x, \mu_y) \tag{1.2}$$

$$S(\mu_x, \mu_y) = \max(\mu_x, \mu_y) \tag{1.3}$$

$$N(\mu_x) = (1 - \mu_x) \tag{1.4}$$

where t-norms (AND operators), s-norms (OR operators, also called as t-conorm) are termed as triangular norms in fuzzy literature and N represents the negation. A short table of widely used t-norm duals in fuzzy control applications is depicted in Table 1.1.

There are three major fuzzy complement operators which have been widely used in the

literature [6]. These operators are;

$$\text{1.Standard Complement:} \quad N(\mu_x) = (1 - \mu_x)$$
$$\text{2.Sugeno's Complement:} \quad N_s(\mu_x) = \frac{1-\mu_x}{1+s\mu_x}$$
$$\text{3.Yager's Complement:} \quad N_w(\mu_x) = (\mu_x \cdot \mu_y)$$

where s is Sugeno's constant and w is Yager's constant.

1.2 Fuzzy Logic Controllers: Principles of Operation

Fuzzy logic controllers (FLCs) primarily depend on the controlled process and the demanded quality of control. It provides a formal methodology to represent human's heuristic knowledge to control a system. By defining these fuzzy controllers, process control can be implemented quickly and easily. For different applications, the control structures vary by the number of inputs, outputs, membership functions, number of rules, and type of inference engine or a method of defuzzification [7, 8]. The choice of these different fuzzy control combinations is in the hands of designer for a particular problem. The most appreciable feature of FLCs is its ability to manage complex control problems through the heuristic rule-of-thumb strategies of the expert provided by fuzzy set theory, instead of using complex differential equations to derive mathematical models of a process plant. This establishes the power of FLCs in nonlinear control plant in recent times [2, 9–11].

Even though there are many analog fuzzy logic controllers in market [12, 13], most of the fuzzy logic controllers have been implemented in digital form. The fuzzy logic controllers discussed in this thesis belong to this group. Hence, the term Binary to Fuzzy (B/F) conversion has been introduced. As inputs of a digital fuzzy controller are defined over discrete universes of discourse with the finite number of elements (integers) obtained after quantization of sensor signals (A/D conversion). The basic structure of fuzzy logic controller is represented in Figure 1.4. It consists of the following modules:

i. Fuzzifier (B/F Conversion): Crisp input data or input data variable to the fuzzy control system is mapped by a sets of the membership function, known as "fuzzy sets." The fuzzification is a process to convert these real variables into linguistic variables or fuzzy variables. The realtime hardware representation of triangular membership function with four MFs is presented in Figure 1.3.

ii. Rule Base: It stores a set of IF-THEN rules, which govern a typical fuzzy system. These

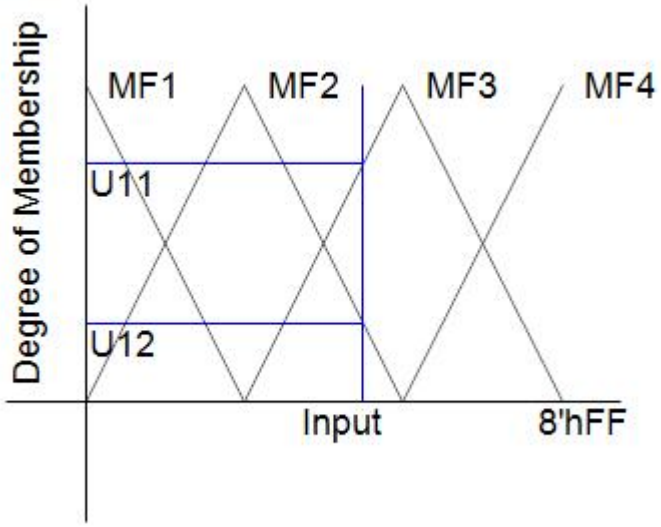


Figure 1.3: The hardware structure of Triangular Membership Function

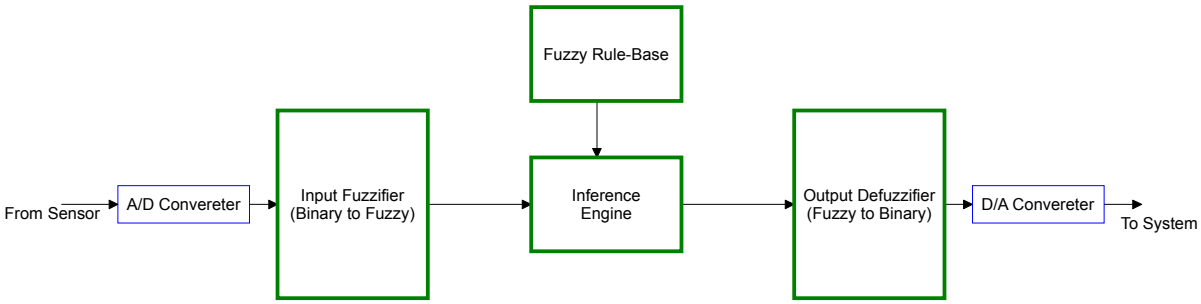


Figure 1.4: The Structure of Fuzzy Logic Controller

rules usually are the expert's linguistic description to achieve good control. These rules describe the output dependence on the inputs, and they are mentioned in terms of the MFs representing the inputs and outputs of the process plant.

iii. Inference Engine: It is a process of identifying rules to calculate the values of the linguistic output variable. The inference step consists of two components:

a. Aggregation: It evaluates the IF part (condition) of the rules.

b. Composition: It evaluates the THEN part (conclusion) of the rules.

iv. Defuzzifier (F/B Conversion): It translates the conclusion of inference mechanism into the substantive crisp controller output or actual inputs to the process plant.

1.3 Hardware Implementations of Fuzzy Logic Controllers

The last two decades have been marked by a great evolution in the field of fuzzy logic to address complex problems of economics [14–17], robotics [18–22], automobiles [23–27], power electronics [28–30], chemical industry [31–34], aerospace [35–40], manufacturing process [41–43], transportation [44–48] and many others [17, 49–54] for their superior performance than the classical control techniques. Fuzzy logic uses the intuitive knowledge and experience of the experts to achieve desired output action. Fuzzy logic provides a formal way to convert this knowledge and experience using IF-THEN rules [55] and makes it a fully structured control algorithm suitable for computer implementations. These factors motivate the engineers to design and implement the fuzzy logic controllers for a wider range of applications. Taking into account of difficulty in different hardware implementations these are categorized by the following type of implementations:

a. Analog fuzzy implementations

b. Digital fuzzy implementations

c. Commercial processer based implementations

Each of these implementations can be further classified based on the target application and aspect of system design. Different forms of FLC implementation is presented in Figure 1.5, where dedicated integrated circuits are designed primarily to target single control applications and built over ASIC with full custom analog, digital, and mixed signals. Programmable integrated circuits based FLCs are developed in integrated circuits (ICs) that can be reconfigured by the user. Commercial processors are using software application to define the FLC system. Bell Labs AT&T has implemented its first digital fuzzy processing [56] device in 1985 that runs at 80,000 FLIPS for a two input one output problem. Later an analog fuzzy processor [57] was built using a bipolar transistor and the processor provided a performance of 1 MFLIPS with defuzzification and 10 MFLIPS without the defuzzification. These two works propelled research in fuzzy implementations using analog and digital hardware with higher processing speed, lower silicon area, and lower power consumption.

1.3.1 Analog Implementations of FLC

One of the major advantages of the analog implementations for fuzzy processing is the absence of Analog to Digital Converter (ADC) and Digital to Analog Converters (DAC)

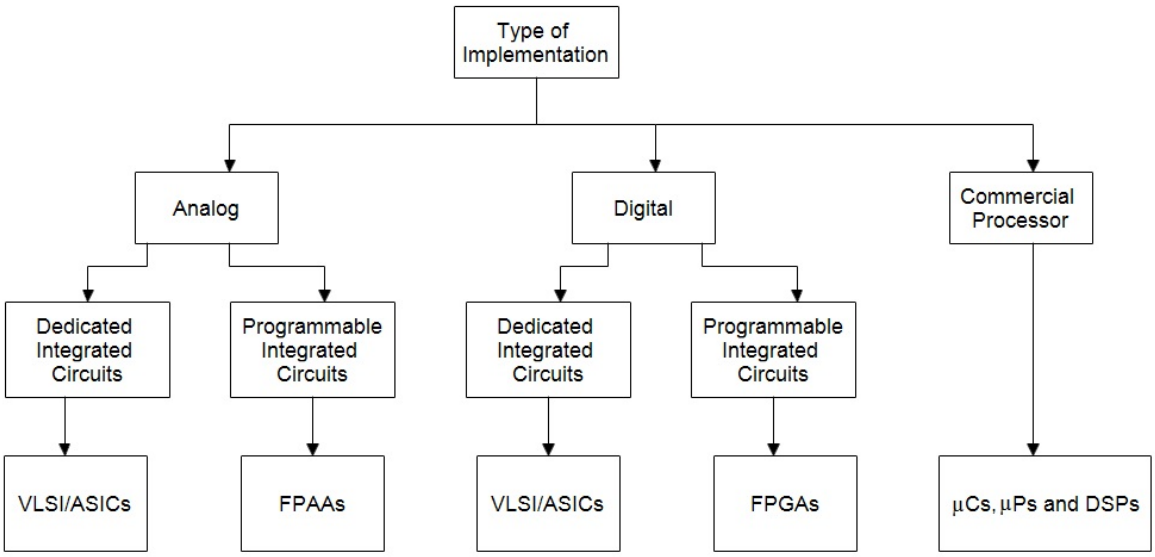


Figure 1.5: Classification of hardware implementations for fuzzy systems

since the analog implementations have a natural connection with different sensors or actuators with either voltage mode or current mode.

1.3.1.1 Dedicated Integrated Circuit based Analog FLCs

There are four modes in which dedicated IC based FLC are implemented:

1. Current mode It is the most suitable architecture for fuzzy basic operations with its advantages like low chip area and low power [58, 59], but suffers from the disadvantage of fan-out limited to '1' and thus can only be connected to a single output.

2. Voltage mode Implementations of FLCs can support more than one input and output. Yamakawa *et. al.* [60] proposed the first device in bipolar technology, which attained the speed of inference engine at 1 μs (1 MFLIPS) and the defuzzification of 5 μs. To achieve higher speed and lower power consumption, Peters *et. al.* [61] implemented analog FLC for the intelligent sensor using CMOS (2.4 μm) technology attaining 2 MFLIPS speed. Marshall and Collins [62], in their design used a floating gate subthreshold technique for FLC with 75 rules and achieved 500 μW power consumption with less than 5 mm^2 area. The potential disadvantage of this design was its low speed of the order of KFLIPS.

3. Transconductance mode Circuits operate in transconductance mode where inputs are in voltage and outputs are in current. Most of the circuits in voltage mode operate in transconductance mode to obtain membership functions. These membership functions

are based on differential pairs of transistors operating in strong or weak inversion [63, 64]. Operational Transconductance Amplifier (OTA) [65, 66] and capacitors for basic blocks are used for the treatment of MFs in other designs.

4. **Switched or circuit discretized mode** incorporate programmability and accuracy in FLCs analog implementations. These designs were introduced in fuzzy controller implementation using switched capacitor techniques [67–69]. Even though these circuits perform well in terms of speed, they had the demerit of high silicon area with a design based on Op-Amps or comparators instead of transistors.

Table 1.2 presents the implementation reference of dedicated IC based FLCs, where special attention has taken to increase the processing speed and restricted fuzzy parameters with a static rule base.

1.3.1.2 Programmable Integrated Circuit based Analog FLCs

FLCs implemented on analog programmable ICs have not attracted engineers significantly, some of the most relevant works with this technique include;

- Pierzchala *et. al.* [70] developed FLC on FPAA usage on multi-valued logic. FLC implemented used m inputs, n rules with trapezoidal membership function.
- Amaraletal *et. al.* [71] relied on Programmable Analog Multiplex Array (PAMA) with GA programming through PCI bus. With this GA code, FPAA was used to configure the membership function.
- Ionita *et. al.* [72] implemented a Mamdani FIS system, an evolutionary algorithm has been used for tuning the MFs.

1.3.2 Digital Implementations of FLC

Fuzzy systems and controls have made fast advancement in past decades. Owing to its widespread usage in consumer electronics and industrial process control, implementation of FLCs has been rigorously researched, and development in terms of implementation has been popular. However, an increase in process complexity of the industrial plants is accelerating demand for controllers with high computational speed, low complexity, easy deployment, and lower development time in terms of design. In order to conform to the demand-supply

Table 1.2: Dedicated IC based analog FLCs

	Application	Year	In	In MFs	Type of In MF	Out	Out Mf	Type of In MF	No Of Rules	Defuzz	Speed	Technology
Current Mode	Max. Operator [58]	1994	2	-	-	1	-	-	-	-	100 ns	CMOS 1.6 μm
	EM Fields [59]	1994	2	-	-	1	-	-	9	-	10 MFLIPS	CMOS 2.4 μm
Voltage Mode	Generic [60]	1993	2	-	Triang,Trapez	1	-	Triang	n	COG	1 MFLIPS	ECL-ECFL
	Automation Sensors [61]	1995	2	7-7	Bell	1	7	-	13	COA	2 MFLIPS	CMOS 2.4 μm
	Generic [73]	1995	3	n-n-n	Triang,Trapez	1	7	-	4	COG	0.6 MFLIPS	BiCMOS 2.0 μm
	Mobile Robot [74]	1996	3	3-3-3	Triang,Trapez	1	5	Singleton	13	COG	6 MFLIPS	CMOS 2.4 μm
	Sensors [62]	1997	3	3-5-5	Triang	1	7	Triang	75	COG	10 KFLIPS	CMOS 2.0 μm
Trans Conductance Mode	MF Generation [66]	1991	3	-	Triang	1	-	-	-	-	82 ns	CMOS (SPICE Sim)
	Max. Input [63]	1994	2/3	-	-	1	-	-	-	-	100 ns	CMOS 1.6 μm
	Generic [65]	1995	2	-	Triang	1	9	singleton	9	COA	15 MFRPS	CMOS 1.6 μm
	Sensors [64]	1996	2	-	bell	5	-	singleton	80	COG	-	CMOS 2.0 μm
Switched Mode	Generic [68]	1993	n	-	S&Z Shaped	-	-	-	-	COG	-	CMOS-
	Generic [75]	1994	2	56	Triang,Trapez	4	28	-	32	COG	16 MFLIPS	CMOS 0.8 μm
	Controllers [67]	1997	4	64	S,Z,Triang & Trapes	2	7	singleton	16	Weighted Average	85 KFLIPS	CMOS 1.2 μm

chain of the industry, FLCs have to be designed accordingly. A unique solution to fulfill this growing market demand is to move to the digital platform. It is well known that digital systems have high resistance to noise, temperature and voltage variations there by making system robust. There are various digital platforms available to design and implementation that reduces quick turnaround time. Although, systems created in digital hardware platforms are not as fast as analog models still, a good system cycle time can be achieved which provides sufficient throughput speed for the majority of the control problems.

1.3.2.1 Dedicated Integrated Circuit based Digital FLCs

FLC implementations on dedicated ICs are concentrated on the structure of fuzzy rules. Also, structure depends on the rules executed in parallel or sequential manner. The subsequent execution of rules used RAM for the storage of rules. Hence the speed is dependent on the parameters like the number of rules, the number of inputs, and number of MFs. In parallel execution, the rules are executed in parallel at the cost of extra LUT for the implementation of MFs with the help of rules stored in ROM. Table 1.3 and Table 1.4 illustrates the work done in these implementations, and these implementations are fixed for a particular application, limited rule base, set to its membership functions, inputs, and outputs. Some of the notable designs include,

- Eichfeld et.al. [76] reported a four-input and single-output FLC with 4096 fuzzy rules with 8 MFs each. However, the system operated only on two overlapping MFs and used singleton type of MFs for output.
- Watanabe et. al. [77] developed FLC in 0.7 μm CMOS process. The system achieved high performance due to its parallel architecture. The FLCs evaluates 64 rules, but this design too used two overlapping MFs method. Also, if four inputs are used the rules are limited by 64.
- Huang et. al. [78] presented a fuzzy inference processor designed in CMOS 0.35 μm process. This design used trapezoidal membership function with fixed rule base.
- Falchieri et. al. [79] designed one of the most flexible structures for FLCs in the literature. This device, however, does not discuss the speed of performance.
- Javadi et. al. [80] design provided a new fuzzification method for hardware on 0.13µm, but it was only applicable to piece-wise linear MFs.

1.3.2.2 Programmable Integrated Circuit based Digital FLCs

In this classification, Field Programmable Gate Arrays (FPGAs) outperforms its predecessor Complex Programmable Logic Devices (CPLDs) since the latter is limited with its logic and function density. Hence there are not many CPLD based FLCs reported in the literature. On the other hand, FPGA provides a number of advantages like re-configurability, short time to market, customization, parallelization, flexibility in design. FLC process is programmed through Hardware Description Language like VHDL (Very high speed integrated circuit Hardware Description Language) or Verilog. Some of the notable developments on FPGAs for fuzzy logic implementations are reported in Table 1.5. Some of the important designs are,

- Hongguo Sun *et. al.* [88] presented a fuzzy PID design on CPLD for PWM trigger pulse generation to a full bridge inverter and a chopper circuit. It implemented a two-input one-output FLCs with fixed rule base and rigid MFs.

- Jingyan Xue *et. al.* [89] presented a novel methodology to design a fuzzy reasoning based expert system on CPLD for fault diagnosis. Similar to the previous design, this too implemented FLCs with fixed rule base and rigid MFs.

- Adhavan *et. al.* [90] countered the problem of a non-uniform variance of the torque developed in a vector controlled permanent magnet synchronous motor by introducing an FIS implemented on an FPGA. The authors have reported that the heuristic knowledge-based fuzzy logic control system (FLCS) has reduced the torque ripple to 1.81%.

- Benzekri *et. al.* [91] reported PD approximated FLCS developed on Cyclone II FPGA to control a dual axis sun tracking system. The simple rules developed with human knowledge have been found to be successful in reducing chip count, cost and development time of the controller significantly.

- Santos and Ferreira [92] implemented a multi-state FLCS on Virtex-II FPGA and NI Compact R10−9002 to control servo- pneumatic actuation systems. They showed significant performance gain in terms of the steady state error, overshoot and settling time.

Table 1.3: Dedicated IC based digital FLCs - parallel rules

Application	Year	In	In MFs	Type of In MF	Out	Out Mf	Type of In MF	No Of Rules	Defuzz	Speed	Technology
Generic [81]	1989	4(4bit)	-	Triang,trapez	2(6bit)	-	-	51	COA	580 KFLIPS	CMOS 1 μm
Navigation [77]	1991	4(6bit)	64	Any Shape	2 (6bit)	-	-	51	COG	150 KFLIPS	CMOS 1 μm
Generic [82]	1996	4(4/6 bit)	-	Triang,trapez	2(4/6bit)	-	Triang	-	Any	-	CMOS 1 μm
Generic [83]	1997	4(8 bit)	7-7-7-7 (6 bit)	Any Shape	1 (8 bit)	8 (6 bit)	-	64	COA	86 MFRPS	CMOS 0.7 μm
Area recognition [79]	2002	2 (7 bit)	8-8	Any Shape	1 (7 bit)	128 (7 bit)	-	64	Sugeno	33.3 MFLIPS	CMOS 0.35 μm
Generic [78]	2005	2(8bit)	No Limit	Trapez	1(8bit)	64 (8bit)	Crisp	64	COG	7 MFLIPS	CMOS 0.35 μm
Fuzzification [80]	2013	N	n	Any Shape	m	-	-	-	-	-	CMOS 0.15 μm

Table 1.4: Dedicated IC based digital FLCs - sequential rules

Application	Year	In	In MFs	Type of In MF	Out	Out Mf	Type of In MF	No Of Rules	Defuzz	Speed	Technology
Generic [76]	1992	4(5bit)	7-7(3bit)	Triang,trapez	1(5bit)	7(3bit)	Triang,trapez	4096	COG	-	CMOS 0.6 μm
Generic [84]	1995	256(6bit)	7-7 (3bit)	Any Shape	64 (6bit)	7(3bit)	Crisp	16,384	COG	10 MFRPS	CMOS 1 μm
Generic [85]	1998	4(8 bit)	7-7	Triang,trapez	1(16bit)	255	Any	2401	COG	0.48 MFRPS	CMOS 1 μm
Radar [86]	1998	2(8 bit)	4 (6 bit)	Trapez	1 (8 bit)	4	-	4	COG	100 KFLIPS	CMOS 1 μm
Air Condition [87]	2005	3 (8 bit)	5-4-5	Gaussian	2 (8 bit)	9 (7 bit)	Singleton	25	Sugeno	-	CMOS 0.35 μm

- Messai *et. al.* [93] reported an FLC to seek maximum power point deliverable by a photovoltaic (PV) module using measures of PV voltage and current.

- Schrieber *et. al.* [94] presented an interval type- II FLCS implemented on a Xilinx Spartan 6 FPGA utilizing DSP48AI slices for different linear and non-linear modules.

- Tamukoh *et. al.* [95] reported a new technique of bit shift based fuzzy inference method for efficient digital hardware implementation. They applied the proposed design on a Virtex-II FPGA for a self-organization relationship network.

These designs depict that the realization of FLCs on FPGA development platform is fast and efficient. However, most of these designs are application specific. It is important to realize that the speed achieved by these designs depends on the fuzzy parameters chosen for the particular application. Hence, the fuzzy parameters are fixed in all implementations. Field tunability and the rule reduction other than overlapping membership function are required in these implementations to enable users to change control parameters in real-time and to increase inference processing time. The Xilinx FPGA flow diagram is presented in Figure 1.6, where the design is first captured using a high-level language like Verilog or VHDL. Then the RTL is synthesized by using synthesis tool to create netlist file. Using the user constraint file (UCF) the implementation stage produces bit stream called bit file that is used to configure FPGA.

1.3.3 Generic Fuzzy Processors

Whenever the speed of operation is not critical, designers choose a software programming on general purpose processors. These are straightforward and widespread architectures present low running speed at low cost. Some of the notable designs include:

- Binfet and Wilamowski [104] designed a FLC on 8-bit μC 68HC711E9, from Motorola. The FLC supported three types of MFs Triangular, Trapezoidal, and Gaussian. It presents two defuzzification processes: Zadeh and Takagi-Sugeno (T-S).

- Nhivekar *et. al.* [105] proposed an FLC for a temperature control system using a μC Atmega8 from Atmel. They considered a single input and single output problem with the number of input MFs (Triangular) as 5, the number of output MFs (Triangular) as 5 and weighted average defuzzification method.

Table 1.5: Programmable IC based digital FLCs - FPGAs

Year	Application	Inputs	Input MFs	Outputs	Output MFs	No Of Rules	Defuzzifier	Speed	Device
1995	Controller [96]	2(6 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	1.67 MFLIPS	SC4008
1996	CAD Truck Control [97]	2(8 Bit)	5-5 (Triang)	1 (8bit)	5(Triang)	11	MOA	1.25 MFLIPS	XC4006
2001	Car Parking [98]	2(8/10/12 bit)	5-7 (Trap/Triang)	1 (8/10/12 bit)	7(singlton)	35	WAM	333/277/222 KFLIPS	FLEX 10K
2006	C0-Processor [99]	2(8 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	2.5 MFLIPS	XC3S200E
2006	General [100]	2(6 Bit)	7-7 (Triang)	1 (6bit)	5(singlton)	9/49	MOM	2.85,0.92 MFLIPS	XC2S200E
2006	Climate Control [101]	4(12 Bit)	7-7 (Triang)	2(12bit)	7(Triang)	16	COA	77 KFLIPS	A54SX32A (Actel)
2008	Bit serial Arthmatic [102]	2(6 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	5.26 MFLIPS	EP1S80B956C6
2010	Mobile Robots [103]	2(12 Bit)	9-9 (Trap/Triang)	1 (12bit)	9(singlton)	81	COG	1.2MFLIPS	Spartan 3E
2011	MPPT [93]	2(16 Bit)	5-5 (Triang)	1 (16bit)	9(singlton)	-	COA	-	EP2C35

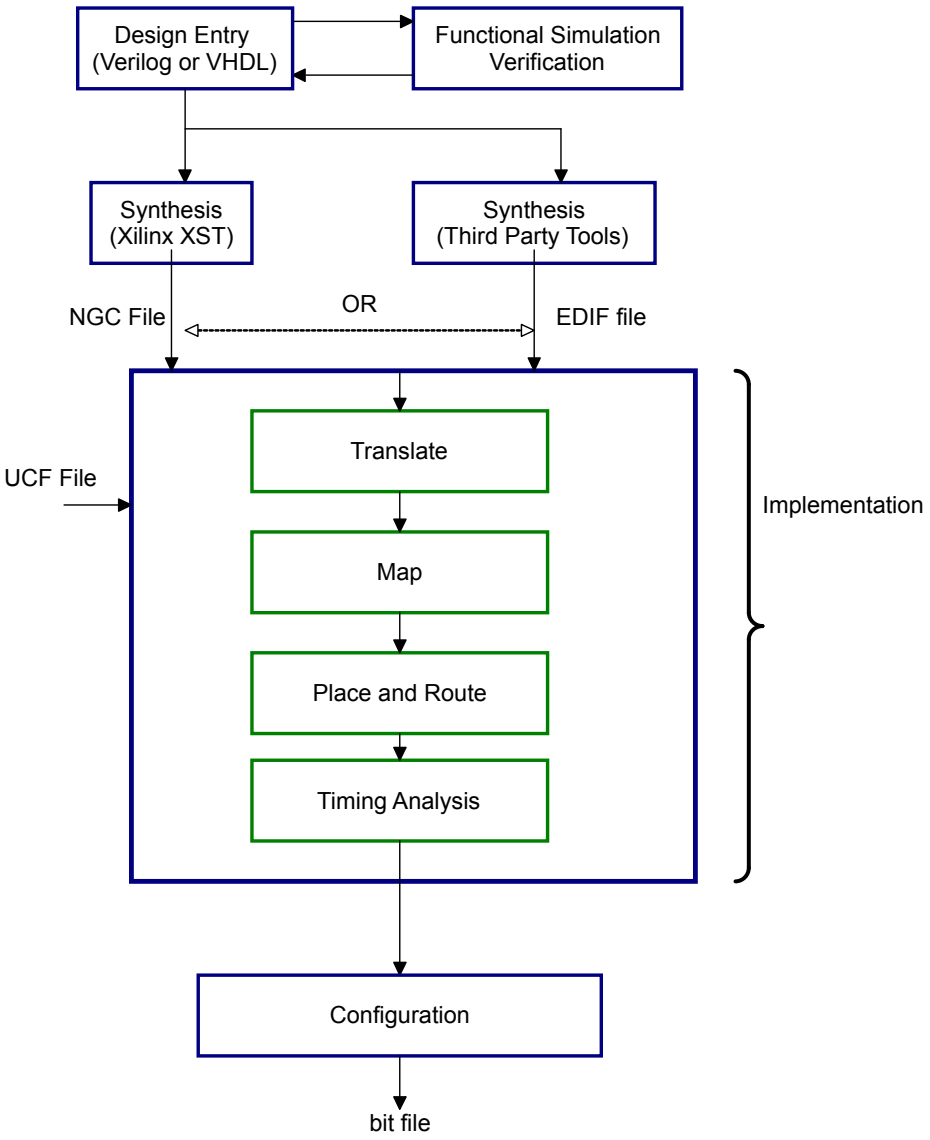


Figure 1.6: FPGA Design Flow

- Eskandarin *et. al.* [106] proposed a fuzzy instantaneous power theory to improve conventional p-q theory dynamic performance and implemented it on a TMS320F28335 DSP device.

- Rafa *et. al.* [107] implemented a new FLCS design on DS1104 DSP to solve coupling problem in vector control of induction motor. With the absence of current regulators this fuzzy vector control of the induction motor provided a low-cost system.

- Pallab Maji [108] designed and implemented generic FLCS as part of his Ph.D. thesis, Where the author developed G-FLCS and fine-tuned the fuzzy parameters with the aid of software framework on the design, which was implemented using a DSP that works with a sequential code.

Table 1.6: Major works on Generic FLCS

Year	Speed (in FLIPS)	Platform	Features				
			Output MFs	I/O	Input MFs	Overlaps	Rules Evaluated
1995	0.63M	BiCMOS 2 μm	Singleton(7)	2-1	11	2	4
1996	48-122	R3000A RISC	-	-	-	2	51
2005	15.87M	CMOS 0.35 μm	Singleton(7)	2-1	3	2	9
2007	16.6M	CMOS 0.35 μm	Singleton(7)	2-1	4	2	16
2008	5.5K	FPGA	Singleton(5)	2-1	8	2	64
2010	11K	FPGA	Singleton(5)	2-1	5	2	25
2011	16.6M	CMOS	Singleton(7)	2-1	4	2	16
2014	15M	CMOS 0.35 μm	Singleton(7)	2-1	5	2	25
2015	NA	CMOS 0.35 μm	Singleton	2-1	4	2	16

- Okumus *et. al.* [109] reported on FLCS design implementation using TMS320F2812 DSP device to control a brushless DC motor and compared the result with HB current controller. The heuristic knowledge based FLC was found to perform extensively well.

- Gai *et. al.* [110] used a TMS320C6713 DSP device to implement a fuzzy based Haar wavelet feature extraction technique to classify successfully and detect a counterfeit banknote.

There are many more DSP based FLC designs that have been successfully implemented in various control applications. It can be readily inferred that the development of DSP-based FLC is easy compared to FPGA [111–113]. However, since the parallel architecture can be implemented on FPGA as compared to sequential processing of DSP, to achieve execution speed in inference process, the FPGA platform is preferred in FLC implementations than the DSP platform.

1.4 Standalone Tunable Digital FLCs

Standalone Tunable Digital Fuzzy Logic Controller (STFLC) systems are standalone and remotely tunable fuzzy logic control devices. These devices are developed on Field Programmable Gate Arrays such that they can be easily interfaced with various process plants. The primary characteristic of this type of hardware includes programmability in run time. These devices accept fuzzy parameters from the users externally through some user interfaces or programmable pins.

STFLC designs are mostly crippled by their operational speed and hence they are generally forced to operate under reduced functionalities. Some of the prime FPGA-based designs from various vendors have been surveyed and tabulated in Table 1.6. The table also

lists the fuzzy parameters reported in these designs. The following observations have been summed up after analyzing these designs.

- It can be observed, that majority of these designs use singleton MFs at the output to reduce computational complexity. Centroid of area (COA) method, when applied to singleton, which is commonly known as weighted average defuzzification method, yields far low computational complexity. However, unlike COA, weighted average does not compute the area under the curve produced from the fuzzy outputs [3]. It can be observed that COA presented in (1.5)

$$Y^* = \frac{\int \mu_c(y) y dy}{\int \mu_c(y) dy} \tag{1.5}$$

  can be reduced to weighted average as depicted in (1.6).

$$Y^* = \frac{\sum \mu_c(y) \cdot y}{\sum \mu_c(y)} \tag{1.6}$$

  where, Y^* represents the crisp output computed from output fuzzy set $\mu_c(y)$ and output support membership function value y.

- These designs use a stringent rule reduction technique, where only two overlapping memberships have been considered.

- These systems evaluate very few rules to improve computational speed. The reduction in the number of rules with only two overlapping membership functions does not provide desired performance in terms of accuracy of the system for many applications.

- In general, these systems cannot be remotely tuned. Some of these devices have MFGs for tuning MFs but the rule base remains static and the performance is limited to two inputs.

These limitations motivate research in hard-core tunable FLC devices on programmable hardware, where multifarious control over the system can be obtained by varying different control parameters with modest computational complexity.

1.5 Motivation of this Work

PID controllers have been widely used in industrial applications even though they are inherently linear and provide a sluggish response. They generally do not provide the

desirable performance to control nonlinear process plants. Their modeling requires a thorough knowledge of system dynamics, and the tuning process is quite complicated. Fuzzy Logic Controllers (FLCs) provide an imprecision based control approach characterized by fast and reliable methods. FLCs architecture can be developed over on Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA), and general purpose processors. Of these, the fuzzy digital hardware based on FPGA becomes the most significant development due to

1. Its ability to reconfigure or reprogram
2. High integration density
3. Parallel structures
4. Low time to market

With these features, the execution time can be dramatically reduced allowing FPGA-based FLC to reach the level of performance of their analog counterparts without their drawbacks (parameter drifts, lack of flexibility). Besides, an FPGA-based FLC can be adapted in runtime to the needs of the plant by dynamically reconfiguring it. Hence, the main motivation of the work is to develop DFLC system with reduced complexity for real-time system.

1.6 Objective of this Work

Based on the motivation for the work, the objectives of the thesis can be considered as follows:

- The primary objective of this research is to design and realize a standalone tunable fuzzy logic controller system on a programmable hardware that can be used for a variety of applications without offline reprogramming.
- Development of a configuration memory map to support the specifications of FLC on flexible hardware.
- Designing the FLC system with the maximum configurable FLC parameters and incorporating fine tuning of fuzzy parameters while the system is in operation.

- Development of rule reduction techniques with their effect on final latency of FLC.
- FLC parameters which need tuning are mainly input range, output range, the number of inputs and number of outputs, the parameters of the membership function like slope and center point of the triangular membership function, If-Else rule base for fuzzy inference process. The flexibility in the system is provided with appropriate tuning for single-input-single-output system or multi-input-multi-output system.
- To handle rule uncertainties, Type 2 FLC outperforms its predecessor Type 1 FLC. This triggers the development of a standalone tunable Type 2 FLC. The successive approximation based method for type 2 FLC is chosen to improve the overall performance in terms of speed.
- Generation of Genetic Algorithm (GA) based optimized fuzzy rule base and incorporate the flexibility in to the FPGA design. This tunes the FLC parameters and makes the FLC system a self-tuned rule-optimized Multi-Input and Multi-Output (MIMO) Fuzzy Logic Controller.

It becomes a challenging task when run-time tunability and flexibility in operation is combined with the standalone mode of operation. Therefore, the architecture of traditional FLC is required to be altered in such a way that, the data integrity and operational methodology remains consistent even after incorporating the above-mentioned features.

1.7 Problem Statement

The limitations of the existing generic FLC designs, as defined in Table 1.6, motivated the research in developing a hard-core STFLC device on FPGA hardware with online tuning, where multifarious control over the system can be obtained with modest computational complexity. However, this design is extremely challenging owing to following conditions.

- Since the proposed design is expected to command a large number of fuzzy parameters; it is imperative to develop an interactive interface for guiding users to input fuzzy parameters.
- It is also a known fact that human beings are prone to make errors while handling large data. Therefore, an automated system has to be deployed which will extract coarse fuzzy parameters from a large input-output dataset.

- The significant challenges in designing such a system lies in managing an exponentially growing rule base. Therefore, development of a suitable rule reduction technique is required which will generate the desired output consuming minimum cycle time.

- It is observed that the fuzzification module in Type 2 FLCs is computationally quite complex. A desirable Type 2 fuzzifier scheme with low computation time is essential to achieve a dependable system cycle time that is relevant to the majority of control applications.

1.8 Outline of Thesis

The thesis is presented in 6 chapters. Following this chapter on introduction, the remaining thesis is organized as follows:

Chapter 2 presents the VLSI architecture of 4 input 2 output system and describes proposed modified rule active 2 overlap membership function (MRA-2-OMF) rule reduction technique.

Chapter 3 presents Modified rule active 3 overlap membership function (MRA-3-OMF) and Modified rule active 4 overlap membership function (MRA-4-OMF) to improve the control accuracy. It explains the design architecture and develops the backbone of proposed Digital Fuzzy Logic Controllers (DFLCs). The structure includes a MATLAB based user interface for users to program fuzzy parameters in the DFLCs. Special state machine is proposed in hardware to support special case rule base.

Chapter 4 presents Type 2 Fuzzifier based on successive approximation method. Wu-Medal inference method is used here for type reduction. Digital Iterative Type 2 Fuzzy Logic Controller (DIT2FLC) is implemented with the rule reduction and special case rule base support, which were proposed in Chapter 2 and Chapter 3.

Chapter 5 presents rule-optimized Multi-Input and Multi-Output (MIMO) fuzzy logic controller on field programmable gate arrays. The design of membership functions in the rule base is made with the aid of genetic algorithm.

Chapter 6 provides conclusion on research work and provides the scope of future work. The limitations of this research are also elaborated in this chapter.

Chapter 2

VLSI Architecture of Fuzzy Logic Controller with Rule Reduction

Preface

In this Chapter, a new VLSI architecture is presented for a generic FLC limited to quad-input and dual-output with maximum seven fuzzy membership functions for each input. The design approach is based on classical three-stage implementation process which includes fuzzification, rule inference, and defuzzification cores. As per the specifications derived in this chapter, the proposed FLC can take a maximum of four inputs with seven membership functions, the rule base comprises of 2401 (7^4) rules. To minimize the complexity, Two Overlap Membership Functions (2-OMF) rule reduction VLSI architecture is suggested to bind the number of rules to 16 (2^4). A Modified Rule Active 2-OMF (MRA2-OMF) rule reduction technique is subsequently proposed to reduce the rules further. Here, the rule inference is implemented for seeking the maximum frequency of operation for targeted Virtex 5 LX110T FPGA. The simulation results obtained with Xilinx ISIM show satisfactory results for all test vectors. Analysis of design is carried out by the Xilinx ISE environment. The performance of the proposed system has been compared with FPGA resources and number of FLIPS in hardware.

2.1 Introduction

For application purposes, the FLC can be divided into two classes:

a. General-purpose fuzzy processor with specialized fuzzy computations.

b. Dedicated fuzzy hardware for specific applications.

The general-purpose fuzzy processor has been implemented on various platforms, such as:

- Computers (PC or WS)
- Processors (μP, μC, digital signal processor (DSP), or transputers;
- Lookup table (digital memory devices).

The relatively simple architecture and the processing algorithm of the FLC naturally lead to straightforward implementations in dedicated hardware. A desirable FLC hardware has two requirements: flexibility and high speed. The general-purpose fuzzy processor can be implemented quickly with flexibility. But, the architecture provides sub-optimal performance, while the dedicated fuzzy hardware requires long development time and can be used restrictively, but offer high performance in terms of operational speed.

In this Chapter, a basic VLSI architecture of the FLC (Figure 1.4) is presented, which opens up with the limitations of FLCs on FPGA resources. When the FPGA based system is used for implementing the desired FLC, all possible design solution can be tested due to the reusability of the FPGA. Often, a hardware implementation of the FPGA-based system is supported by many existing EDA tools for modeling, synthesis, verification, and implementation. The major advantage of using the EDA tools is that the same hardware description language code for modeling can be directly used for synthesis, verification, and implementation. Also, the general architecture of the FLC is invariant except for the change in the number of input and output variables, the number of fuzzy terms, the type of membership functions and its parameters, the bit resolutions, and the control rule base according to the applications. Therefore, it is essential to create the VERILOG code of the desired FLC from the design specifications. The FPGA-based implementation of controllers can efficiently answer current and future challenges of this field. Among them the following can be quoted,

1. Decrease in the cost of implementation for at least three reasons. The use of an architecture based on the specific needs of the algorithm to implement, the application of highly advanced and specific methodologies for improving implementation time, which is also called as "time to market", and the expected development in VLSI design that will allow integrating a full control system with its analog interface in a single chip, also called System-on-a-Chip (SoC).

2. Confidentiality, a particular architecture of a system cannot be easily duplicable.

3. Embedded system implementing with a large number of constraints in application areas like aircraft applications with limited power consumption, thermal consideration, reliability, and single event upset protection.

4. Improvement of control performance in terms of execution time. The execution time can be dramatically reduced by designing dedicated parallel architectures, allowing FPGA-based controllers to reach the level of performance of their analog counterparts without their drawbacks (parameter drifts, lack of flexibility). Besides, an FPGA-based controller can be adapted in runtime altering to the needs of the plant through dynamic reconfiguring it.

5. FPGA's are the only reprogrammable, digital platforms which combine:

    - Logic design
    - Micro-Processing and computer architecture
    - Digital Signal Processing (DSP)
    - Multi-Gbps communications

6. FPGAs provide I/O flexibility which allows new concepts to be quickly verified

This chapter introduces a novel rule reduction technique Modified Rule Active 2 Overlap Membership Functions. This technique improves the accuracy of existing overlap based rule reduction method. Fast inference time is an important feature of the overlap based rule reduction technique. The proposed system has achieved to increase the accuracy by increasing the inference time.

2.2 VLSI Architecture of FLC

For the practical applications of fuzzy logic system, the following constraints have been applied to reduce hardware complexity of the FLC and improve the FLC's control performance.

1. The observed inputs of the FLC are crisp and can be quantized into a finite number of levels.
2. Symmetric triangular membership function is used at output variable.
3. Both membership values and widths of constituent membership functions are used to compute an accurate crisp value.

The implementation specifications of the FLC are as under:

- Number of input variables (m) = Maximum of 4;
- Number of output variable (n)= Maximum of 2;
- Number of membership functions per input variable (p) = (7 Max);
- Number of membership functions per output variable (q) = (7 Max);
- Range of input and output variables (r) = 8bit/16bit= 256/65536 intervals;
- Discourse of membership functions (d) = 8bit/16bit = 256/65536 intervals;
- Resolution of membership values (R) = 1/256 (for bus width =8) or 1/65536 (for bus width =16);
- Implication and Aggregation Model is mamdani;

The VLSI architecture of the above specified FLC with four inputs and two outputs with the Test Bench (TB) is presented in Figure 2.1. The system comprises of fuzzifier which converts the real world input called crisp input to fuzzy variables. In this architecture 8/16 bit (the bus width is parameterized) bus line is used to supply the inputs. There are four fuzzifier blocks for each of the four inputs. The input fuzzified variable is given to rule base module with interface control signals. Rule base uses Min-Max Inference mechanism. The detailed design procedure for rule base module is presented in this section. Defuzzifier uses

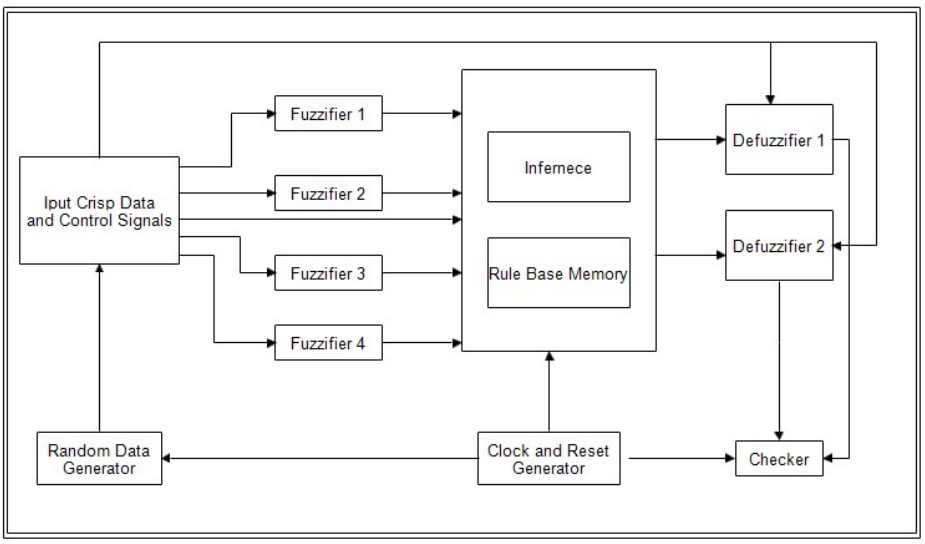


Figure 2.1: The Architecture of FLC in RTL (Register Transfer Level)

the rule base output with centroid method to provide crisp data as output. For testing, fuzzy input data is generated randomly by random generator and the checker checks the defuzzified output with expected output.

The design of each module in FLC is elaborated in this section with its pin diagram, pin description, necessary equations, and logical timing diagrams.

2.2.1 Fuzzifier

The fuzzifier module computes membership values for given input, the membership function for each input. The fuzzification operation involves calculating the abscissa for a given line and ordinate. It returns membership values between 0 and 1 for crisp input values provided in any range. Therefore, a divider is designed and implemented to find the slope (μ) from (2.1). The divider uses a faster version of the non-restoring division algorithm involving shifting the divisor to the right while checking the difference with the dividend. The fuzzification block accepts an 8/16 bit crisp digital data and generates fuzzified output each of 8/16 bit width corresponding to the membership functions used. There are different forms of memberships functions associated with each input and output response. In this design symmetrical triangular membership function has been used to define inputs and outputs to reduce hardware complexity.

The fuzzifier checks if the inputs are in range for calculation. If it lies beyond the extremes of the triangle, then the membership values are calculated as zero as shown in Figure 2.2. Else the output would be calculated using the divider as per the following

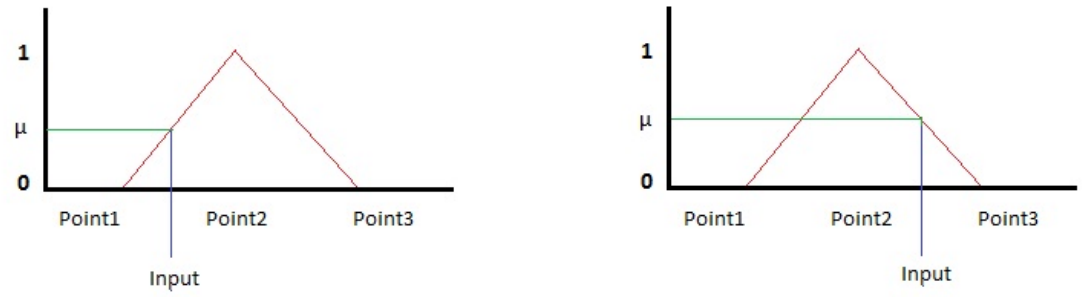


Figure 2.2: Calculation of membership values

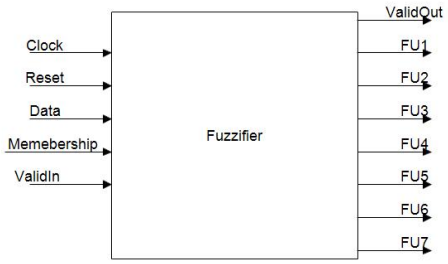


Figure 2.3: Pin Diagram of Fuzzifier

equations,

If Input$\geq$ Point1 and Input $\leq$ Point2

$$\mu = \frac{Input - Point1}{Point2 - Point1} \tag{2.1}$$

And if Input $\geq$ Point2 and Input $\leq$ Point3

$$\mu = \frac{Input - Point2}{Point3 - Point2} \tag{2.2}$$

Pin details for the fuzzifier block are presented at Figure 2.3, where inputs and outputs are labeled. The input-output pin descriptions are described in detail in Table 2.1.

2.2.2 Rule Base and Inference Engine

The block model of the rule evaluator along with fuzzifier output is shown in Figure 2.4 for Mamdani Inference Engine. A two input system design is used to explain the operation here, for its simplicity. The rule evaluator model is the second stage of the fuzzy logic model. This block provides the decision on the inputs to which output is to be chosen. This design consists of two memories. One memory is user programmable where a user can program the rule base using expert experience in that particular application, and another memory to

Table 2.1: Pin Description of Fuzzifier

Pin Name	Direction	Size	Description
Clock	Input	1	Synchronous Clock
Reset	Input	1	Asynchronous Reset
ValidIn	Input	1	Valid Signal for input crisp digital data
Data	Input	8/16	8 or 16 bit Crisp Data
Membership	Input	56/112	56/112 bit data for 7 membership functions with 8/16 bit size each
ValidOut	Output	1	Valid signal for fuzzified data
FU1	Output	8/16	Fuzzy data for membership degree 0
FU2	Output	8/16	Fuzzy data for membership degree 1
FU3	Output	8/16	Fuzzy data for membership degree 2
FU4	Output	8/16	Fuzzy data for membership degree 3
FU5	Output	8/16	Fuzzy data for membership degree 4
FU6	Output	8/16	Fuzzy data for membership degree 5
FU7	Output	8/16	Fuzzy data for membership degree 6

store the minimum values of corresponding seven membership functions of two inputs. In this design, memories have been chosen to utilize the memory blocks in the FPGA and to reduce the logic count and corresponding delay and power dissipation. The two input fuzzy inference system has 49 rules, since two inputs each having seven membership functions translates to 7^2 combinations of rules. From the enable signal of fuzzifier, there is an address generator block where it reads data from rule memory. The Max Function finds the maximum value out of all minimum values stored in minimum finding memory. This function runs in parallel for all seven output membership functions. Here 49 clocks are consumed for the address generations and corresponding memory reading. The two input one output Inference Engine pin details is presented in Figure 2.5. The physical structure of the fuzzy rule base is expressed as

$$R^l : IF\ X_1\ is\ M_1^l\ and\ X_2\ is\ M_2^l ... and\ X_n\ is\ M_n^l\ THEN\ Y\ is\ N_l \tag{2.3}$$

Where, X_1, X_2, ... X_n are the input variables and Y is the output variable. M_n^l and N_l are linguistic values represented as fuzzy subsets of the respective universe of discourse U_i and V at input and output respectively.

For a four input two output system, the input output pin details of Inference Engine are

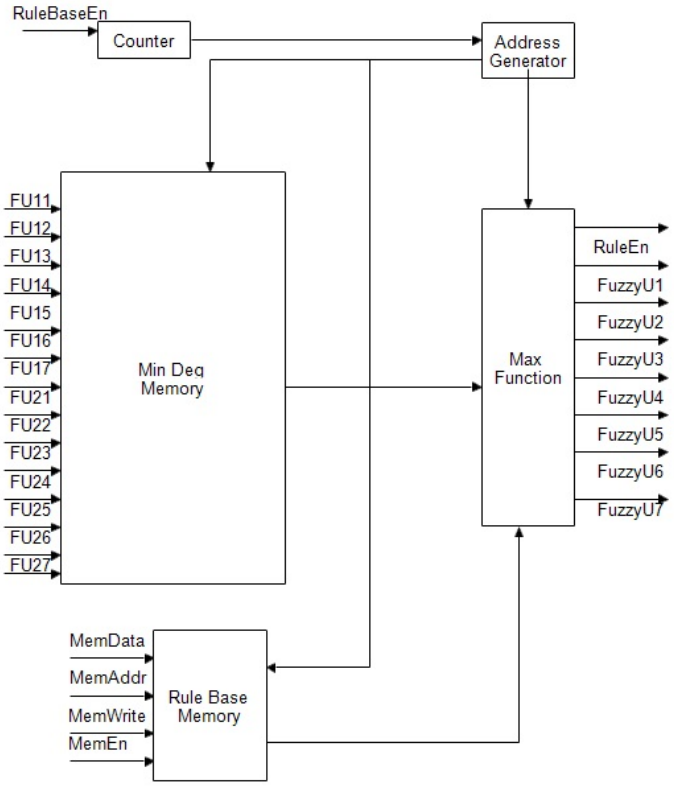


Figure 2.4: Block Model of Mamdani Inference Engine

shown in Figure 2.6. The number of rules used in this design is 4^7 = 2401 rules. The choise of Sugeno Inference Engine (SIE) for inference engine model is also examined, Where SIE hardware needs more parameters to program and ultimately SIE consumes more area than mamdani inference engine. Hence in this thesis all solutions are analyzed using Mamdani Inference Engine.

2.2.3 Defuzzifier

The defuzzifier converts fuzzy inference output values into the crisp output. There exist different methods for defuzzification. The center of gravity method has been chosen for this work. Defuzzifier consists of fuzzy mean module and division module. Multiplication and addition operation are computed in fuzzy mean module and division operations are implemented by division module. The inputs to the fuzzy mean module are rulebase fuzzy output set and membership elements. The centroid method is also known as the "center of gravity" or "area method". It obtains the center of area (x^*) occupied by a fuzzy set. The relationship to find the x^* is given by

$$x^* = \frac{\sum_{i=1}^{n} x_i \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)} \tag{2.4}$$

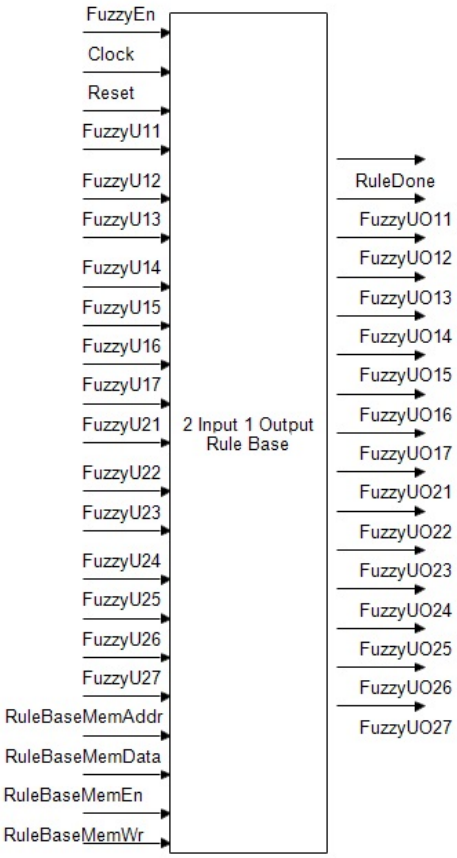


Figure 2.5: Pin Diagram of 2 input 1 output Rule Evaluator

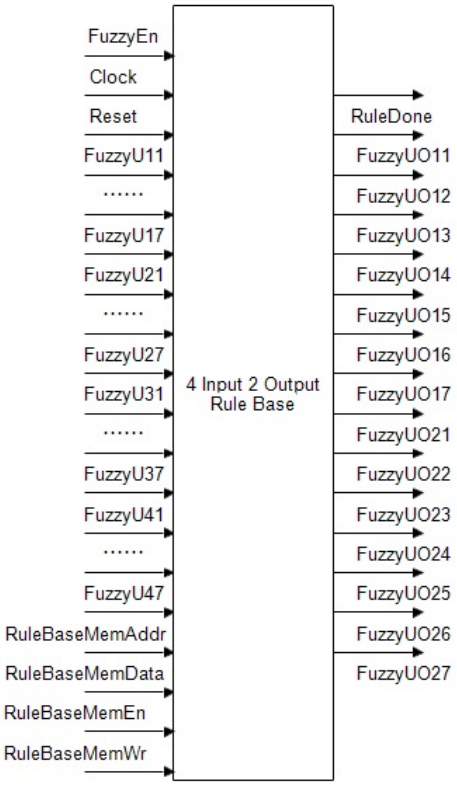
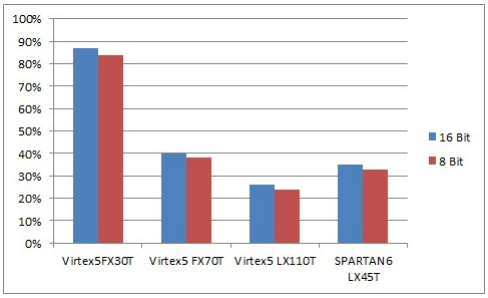


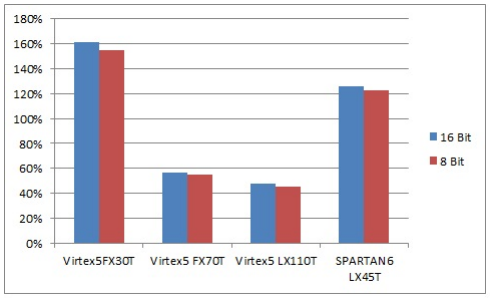
Figure 2.6: Pin Diagram of 4 input 2 output Rule Evaluator

2.3 FPGA Utilization Analysis

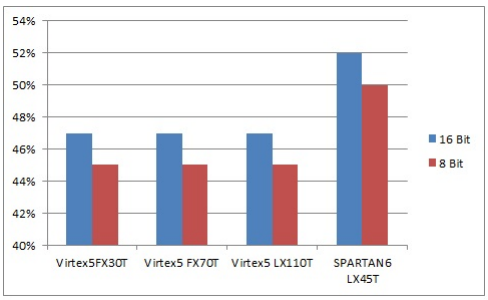
The developed architecture of an FLC is coded in VERILOG Hardware Descriptive Language. Different Xilinx FPGAs are compared for logic utilization in Figure 2.7, here slice registers utilization is presented in Figure 2.7a, Slice LUTs utilization is in Figure 2.7b and fully used LUTs utilization in Figure 2.7c. From these graphs, it can be readily observed that the area utilization of different FPGA's is limited to 50%. It is also very hard to add software interface to this FLC with the available logic left, since these soft cores occupy at least 50% of FPGA logic. So here is a need for rule reduction algorithm to be implemented. So that, the four inputs system with seven membership functions can be reduced from 4^7 =2402 rules to manageable amount. This can further reduce the memory utilization, number of min, max functions inside the hardware thereby leading to the lower resource utilization. In the following section, modified VLSI architecture is proposed to minimize the resource utilization.



(a) Percentage Utilization of Slice Registers in different FPGAs values



(b) Percentage Utilization of Slice LUTs on different FPGAs



(c) Full Utilized LUT Flip Flops on different FPGAs

Figure 2.7: Logic Utilization of FLC on different FPGAs

2.4 2-Overlap Membership Function (2-OMF) Rule Reduction

Once membership functions are defined for input and output variables in an FLC, a control rule base can be developed to relate the output actions of the controller to the observed inputs. This phase is known as the inference or a rule definition portion of the fuzzy logic. For a system with m = number of membership functions and N = number of inputs, N^m rules can be created to define the actions of the fuzzy logic controller. This section discusses VLSI architecture for reduced rule base module for a generalized fuzzy logic controller with the specifications identified in the earlier section.

2.4.1 2-OMF Method and its VLSI Architecture

2.4.1.1 Rule Reduction using 2-OMF method

Majority of the fuzzy logic inferences uses Mamdani model [114–116]. Where, all FLC modules use conventional search space for the rules, which make the system performance slow down. Kalaykov *et. al.* [117] states that the 2-OMF significantly reduces the search space for rules which contribute the FLC output. The condition for this 2-OMF method is that the system parameters should be defined in such a way that no more than two overlaps occur in input search space. However, this rule reduction method is constrained by anticipating fixed number of overlaps that affects the controller performance. It is also characterized by a layered parallel architecture of the fuzzy inference. Moreover, it reduces the dependency of processing time on the number of inputs of the fuzzy system while dependence on the number of rules and fuzzy partitioning of all variables are completely eradicated. A 4 input FLC is considered, Where the input variable $x(t) = [x_1(t), x_2(t), x_3(t), x_4(t)]$ are fuzzified them using membership functions results in the following matrices. The degree of membership

$$U(t) = \begin{bmatrix} U11 & U12 & U13 & U14 & U15 & U16 & U17 \\ U21 & U22 & U23 & U24 & U25 & U26 & U27 \\ U31 & U32 & U33 & U34 & U35 & U36 & U37 \\ U41 & U42 & U43 & U44 & U45 & U46 & U47 \end{bmatrix}$$

The fired fuzzy set indexes

$$I(t) = \begin{bmatrix} I11 & I12 & .... & I1N_1^{Over} \\ I21 & I22 & .... & I2N_2^{Over} \\ I31 & I32 & .... & I3N_3^{Over} \\ I41 & I42 & .... & I4N_4^{Over} \end{bmatrix}$$

Where N_i^{Over}, i=1, 2, 3, 4 is the number of overlaps in membership functions. The Inference mechanism derives the control action according to the fuzzy rule base

$$R_J : if\ x_1 = I_{1J} \text{ and } x_2 = I_{2J} \text{ and } x_3 = I_{3J} \text{ and } x_4 = I_{4J} \text{ then } y = V_J \tag{2.5}$$

Where, V_J = [V_1, V_2, V_3, V_4, V_5, V_6, V_7] is the output term set for control action. $x(t)$ forms the input space and $x_1(t)$, $x_2(t)$, $x_3(t)$, $x_4(t)$ are the linguistic values of the input space. The maximum number of cells for a generalized system, where each cell corresponds to a linguistic value of the control action.

$$N_{cells} = \prod_{i=1}^{n} N_i^{Over} \tag{2.6}$$

Where, n = Number of crisp inputs. So the maximum number of rules that the system defined in section 2.2, with 4 inputs and 7 maximum overlaps has rule dimension of N_{cells} = 7^4 = 2401 rules. If the uncertainty has the boundary between two sets the resulting rule dimension is N_{cells} = 2^4 and the other points computation is not essential. This scenario is presented in Figure 2.8, When the crisp inputs vary (Values x_1^* and x_2^*), it fires at most two fuzzy sets and results 2-OMF cannot have more than four rules. Hereby, 2-OMF will take values from first column of Table 2.2 depending on the system inputs.

Table 2.2: Computed N_{cells} with varying n and Overlaps

	$n=1$	$n=2$	$n=3$	$n=4$
$Overlaps = 2$	2	4	8	16
$Overlaps = 3$	3	9	27	81
$Overlaps = 4$	4	16	64	256
$Overlaps = 5$	5	25	125	625
$Overlaps = 6$	6	36	216	1296

The processing of all rules as presented in section 2.2 is not essential, since these rules have negligible contribution to the final result and consume considerable computation time.

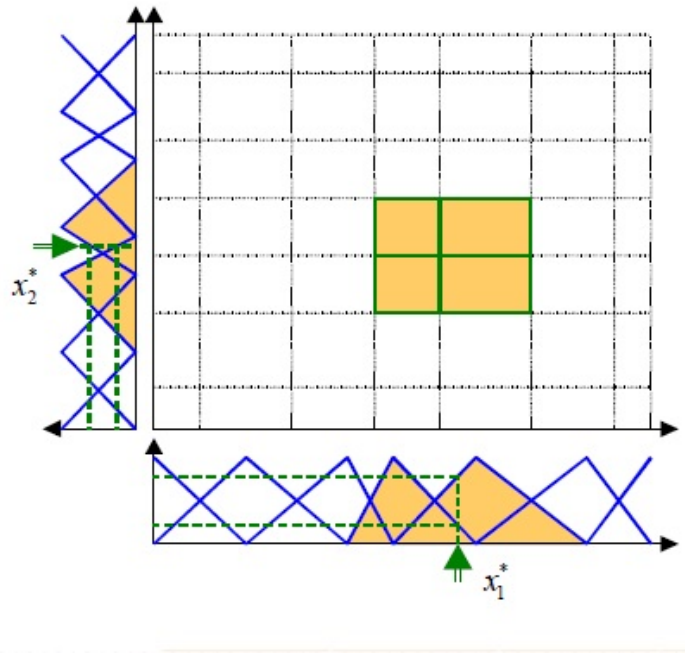


Figure 2.8: 2-OMF Rule reduction concept

Hence, the VLSI architecture is modified according to the 2-OMF method by limiting the fuzzy sets participating in the inference.

From (2.6) a set of N_{cells} containing up to n antecedents usually characterize the fuzzy output. If all the variables are present in each rule, the maximum number of fuzzy rules can be derived is

$$N_{total} = \prod_{i=1}^{n} N_i^{Over\_Max} \tag{2.7}$$

where $N_i^{Over\_Max}$ = number of maximum membership functions, For 2-OMF $Over_Max = 2$ and the maximum number of 2-OMF rules can be derived is

$$N_{2OMF} = \prod_{i=1}^{n} N_i^2 \tag{2.8}$$

The fraction F_{2OMF} of 2-OMF rules with total rules is

$$F_{2OMF} = \frac{\prod_{i=1}^{n} N_i^2}{\prod_{i=1}^{n} N_i^{Over\_Max}} \tag{2.9}$$

2.4.1.2 VLSI Architecture

Once membership functions are defined for input and output variables, a control rule base can be developed to relate the output actions of the controller to the observed inputs. This

phase is known as the inference or a rule definition portion of the fuzzy logic. This section presents the VLSI Architecture of the 2-OMF reduced rule base module for a generalized fuzzy logic controller with the specifications defined in section 2.2.

The modified VLSI architecture is presented in Figure 2.9 includes the following five principal units:

- Rule selector unit selects non-zero fuzzy term set.
- An address generator unit generates an address to read the appropriate rule.
- Rule base memory unit stores the user defined rules. run-time programmability is supported to program rulebase at any interval of time.
- An inference engine unit which performs with approximate reasoning by associating input variables with fuzzy rules. The inference engine unit performs mamdani min-max implication operation and sugeno inference mechanism. It calculates the degree of applicability of all rules selected from the rule base memory by the address generator.
- The CPU registers unit is used to store fuzzified values for four input variables, inference output values, and register to program number of inputs and number of membership functions (tuning parameters).

The results are stored in the CPU registers for defuzzification process. Subsequently, a Finite State Machine (FSM) as shown in Figure 2.10 within the FPGA controls the entire process for all modules.

2.4.1.3 Design Choices for Internal Modules

As discussed in the section 2.2, the rule base is filled with 2401 rules, and the user loads this data in the addresses 0 to 2400. To read the appropriate rule for non-zero fuzzified values it needs to generate matching addresses with its index numbers.

The rule index values segregated from the rule selector for each non-zero membership value are directed to the module Rule Address Generator. This module generates the rule addresses from the index number, where the address is made with index binary numbers 000, 001, 010, 011, 100, 101, and 110. The binary value "111" is an invalid number since the maxim membership functions programmed are 7. The module generates an 'InvalidAddress'

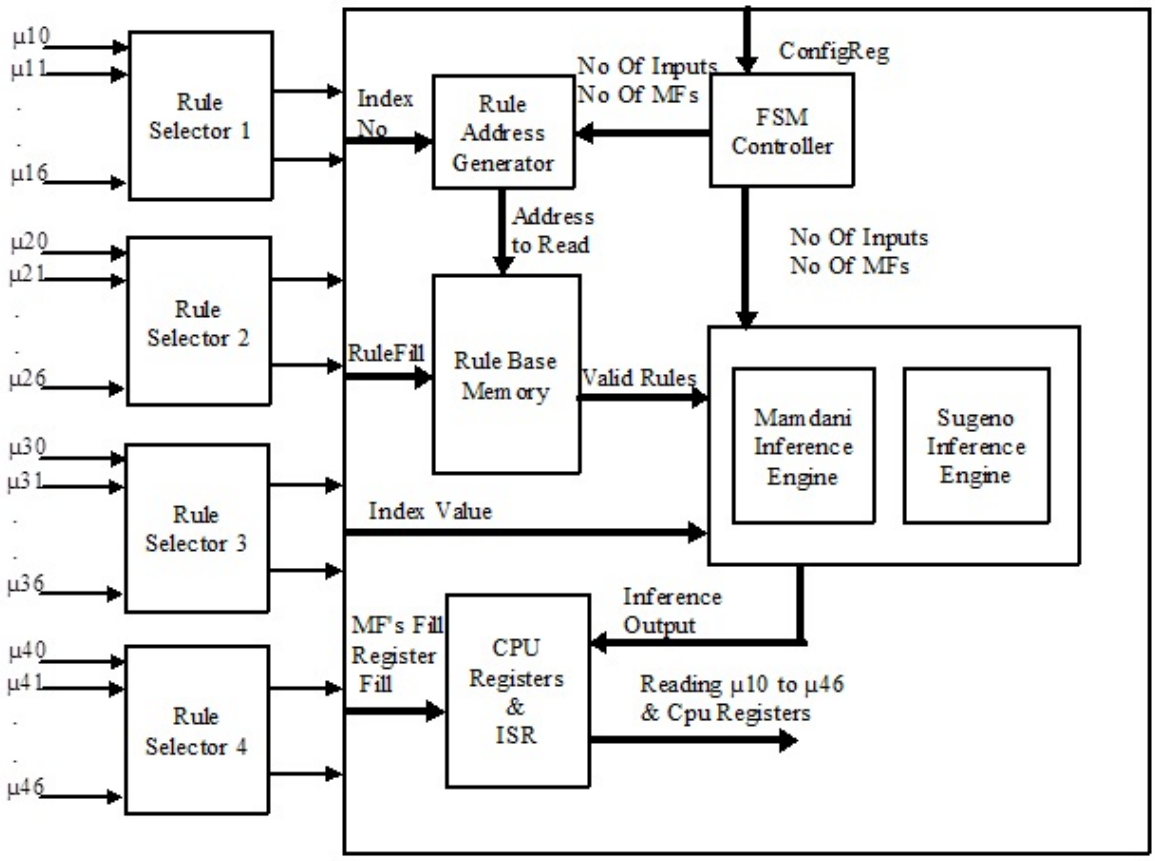


Figure 2.9: VLSI Architecture of 2-OMF Reduced Rule Base

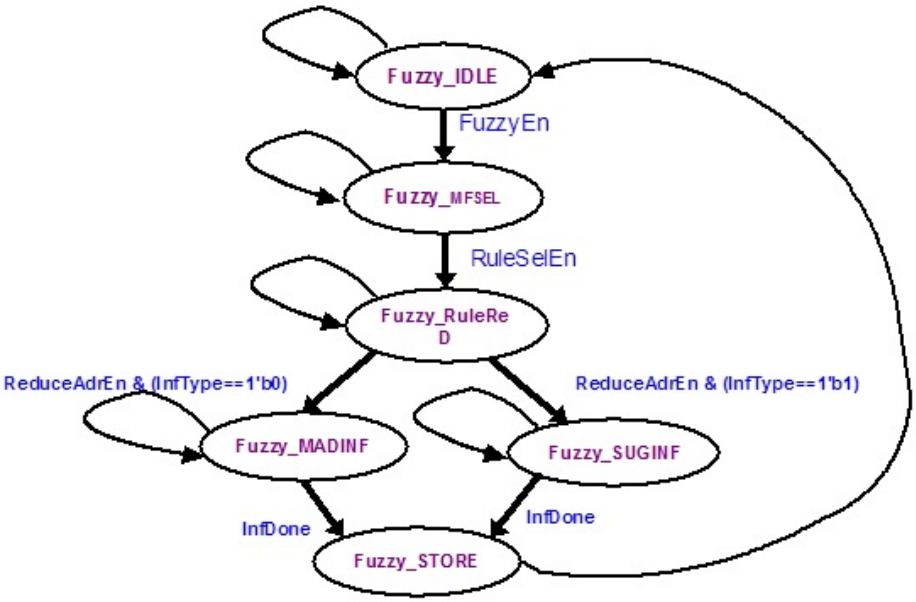


Figure 2.10: Finite State Machine

	Index 4	Index 3	Index 2	Index 1	Rule Address Decimal Value
Rule 1	101	100	010	001	2833
Rule 2	101	100	010	010	2834
Rule 3	101	100	011	001	2841
Rule 4	101	100	011	010	2842
Rule 5	101	101	010	001	2897
Rule 6	101	101	010	010	2898
Rule 7	101	101	011	001	2905
Rule 8	101	101	011	010	2906
Rule 9	110	100	010	001	3345
Rule 10	110	100	010	010	3346
Rule 11	110	100	011	001	3353
Rule 12	110	100	011	010	3354
Rule 13	110	101	010	001	3409
Rule 14	110	101	010	010	3410
Rule 15	110	101	011	001	3417
Rule 16	110	101	011	010	3418

Figure 2.11: Rule address before mapping

signal if one of index value is equal to the binary value "111". 'NoOfInputs' signal is used to find number of addresses to be generated. If 'NoOfInputs' is four, the address generated are 16 else if 'NoOfInputs' are three, then the address generated are 8, and for two inputs addresses are 4. If the inference is of Sugeno model address generation depends on the 'Ready' signal generated in FSM.

Let us assume that a fuzzy rule contains 2-OMF values of input x_1 fired with fuzzy sets I_{11} and I_{12}, x_2 fired with fuzzy sets I_{22} and I_{23}, x_3 fired with fuzzy sets I_{34} and I_{35}, x_4 fired with fuzzy sets I_{45} and I_{46}. To read the corresponding consequent of the rule, the rule addresses are generated as in Figure 2.11. Address generator module does the mapping of rule addresses received from rule address generator module to exact memory location which is from address 00H to address 960H. At each stage, the addresses are generated by the input configuration values. The circuit to generate these addresses is presented in Figure 2.12. In this circuit, based on the number of inputs programmed as 1, 2, 3, or 4 the corresponding DATAA, DATAB, DATAC, or DATAD of MUX is selected to make the reduced rule address. The mapping of addresses using index address with rule base memory address is shown in Figure 2.13.

It is evident from this design that the rule search space for Fuzzy Inference Engine (FIE) using 2-OMF is reduced drastically from N_{total} rules to N_{2OMF} rules. There are two significant drawbacks identified in 2-OMF method. The first one is in its extraction when the total number of rules programmed is less than 2-OMF rules. Let us assume that the complete rules scheduled are P and $P < N_{2OMF}$ then the unnecessary rules that are part in FIE are N_{2OMF}–P. These extra rules increase the latency by N_{2OMF}–P clocks and resulting in

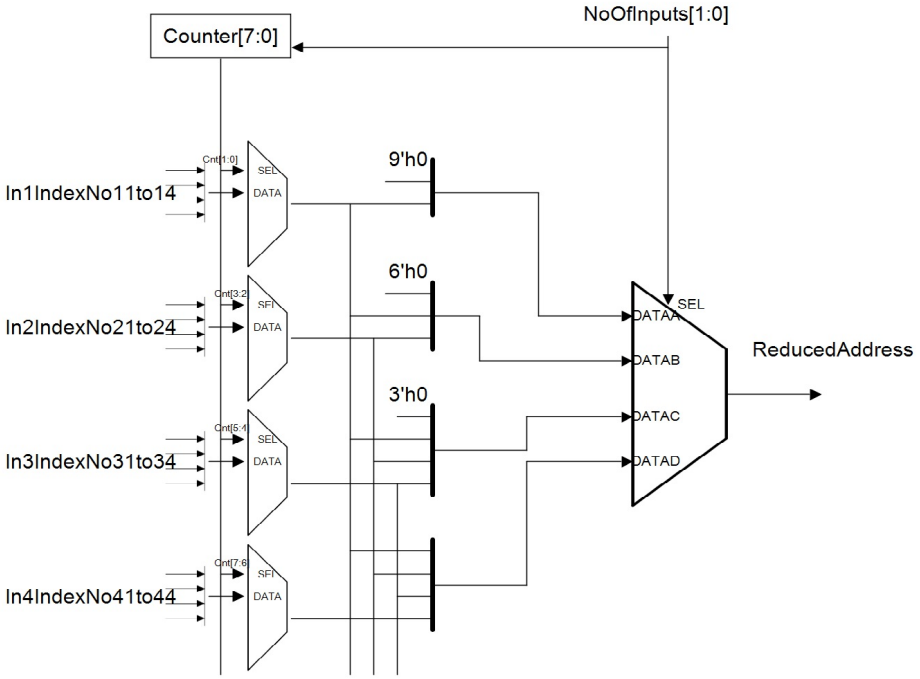


Figure 2.12: Circuit diagram of rule address generator for 2-OMF method

Index Address				Rule Base Memory Address
IxNo4	IxNo3	IxNo2	IxNo1	
000	000	000	000	000000000000 (0)
000	000	000	001	000000000001 (1)
.	.	.	.	.
.	.	.	.	.
000	000	000	110	000000000110 (6)
000	000	000	111	Invalid Address
000	000	001	000	000000000111 (7)
000	000	001	001	000000001000 (8)
.	.	.	.	.
000	000	110	110	000000110000 (48)
000	000	110	111	Invalid Address
.	.	.	.	.
000	000	111	111	Invalid Address
000	001	000	000	000000110001 (49)
000	001	000	001	000000110010 (50)
.	.	.	.	.
000	110	110	110	000101010111 (342)
000	110	110	111	Invalid Address
.	.	.	.	.
000	111	111	111	Invalid Address
001	000	000	000	000101011000 (343)
001	000	000	001	000101011001 (344)
.	.	.	.	.
110	110	110	110	100101100001 (2400)
110	110	110	111	Invalid Address
.	.	.	.	.
111	111	111	111	Invalid Address

Figure 2.13: Rule Address Mapping

slower performance. The second one is, when $P \geq N_{2OMF}$ in specific cases these N_{2OMF} rules may not contribute in the inference. Modified Rule Active 2-Overlap Membership Function (MRA2-OMF) is proposed in the next section to identify these conditions and reduce the rule search space further. Next section elaborates this concept more in detail along with its implementation details.

2.5 Modified 2-Overlap Membership Function With Rule Active (MRA2-OMF) Rule Reduction

In the design of FLC architecture, it is analyzed the inference processing in stages. This analysis optimizes inference processing times and guides the designer in choosing the inference architecture. The analysis is presented in two directions. The first, application of 2-OMF method on inference processing once the input variables are fixed. The second, thrust is identifying the rules that make a contribution to the output (rules that are active). Usually, these rules constitute a small percentage of the complete rules but can make their impact on processing speed. The combination of 2-OMF rule reduction with active rules led to the definition of a computational model that reduce the search space of rule set there by reduces the computational time.

If I_K^i is the consequent derived from rule R_K by truncating conclusion, I_K at θ_k (the k^{th} rule degree of truth), can obtain consequent I. I results from applying all the rules throughout the union of all I_K^i with $1 < K < N_{total}$. If the degree of activation θ_k is null, I_K^i is also null, i.e. in each element, the degree of membership is null in the universe of discourse. Consequently, all these rules with degree of null activation do not contribute to the final fuzzy set I. The number of these rules is

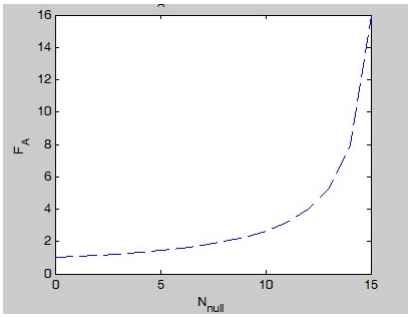
$$N_{MRA2-OMF} = \prod_{i=1}^{n} N_i^2 - N_{null} \tag{2.10}$$

The fraction F_A of MRA2-OMF rules with 2-OMF rules is

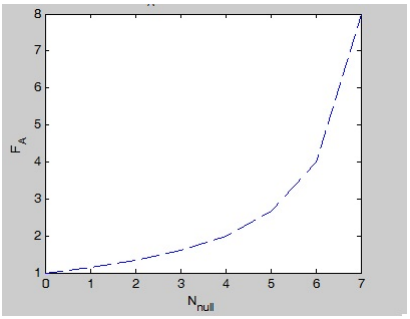
$$F_A = \frac{\prod_{i=1}^{n} N_i^2}{\prod_{i=1}^{n} N_i^2 - N_{null}} \tag{2.11}$$

and the fraction F_{RA} of MRA2-OMF rules with total rules is

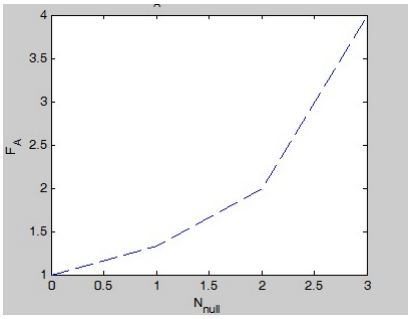
$$F_{RA} = \frac{\prod_{i=1}^{n} N_i^{Over}}{\prod_{i=1}^{n} N_i^{Over} - N_{null}} \tag{2.12}$$



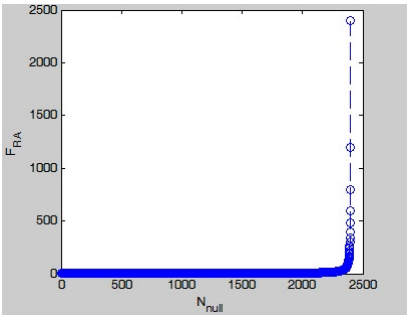
(a) F_A for 4 input variables



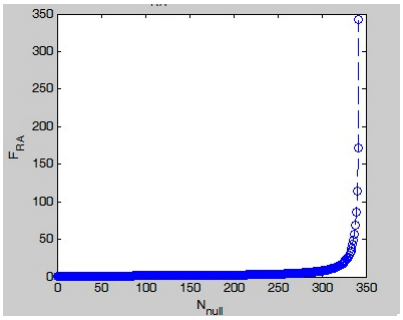
(b) F_A for 3 input variables



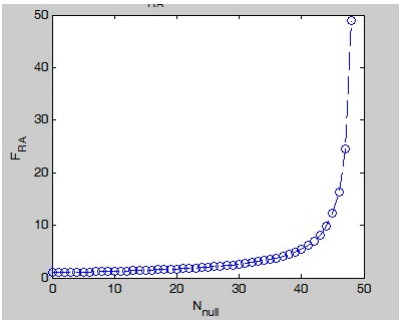
(c) F_A for 2 input variables



(d) F_{RA} for 4 input variables



(e) F_{RA} for 3 input variables



(f) F_{RA} for 2 input variables

Figure 2.14: MRA2-OMF method fractional values with different input variables.

According to the specification, with 4 inputs, each input with a fuzzy set of 7, the F_{2OMF} equals to 0.0066 and F_A, F_{RA} are presented in Figure 2.14 for each N_{null}. Where Figure 2.14a, Figure 2.14b, Figure 2.14c, and Figure 2.14d, Figure 2.14e, Figure 2.14f are plotted for the configurable input parameter values 4, 3, and 2 respectively. Even when a fuzzy process does not comprise all the rules, the number of MRA2-OMF rules reduces to a much lower numbers. Thus processing all the rules is not necessary, because these rules make no contribution to the final result, and they consume computing time leading to slow performance.

Figure 2.15 presents the modified design of the reduced rule generation by considering active rules fired from the rule set. In the case of 2-OMF it is found that there were 16 rules,

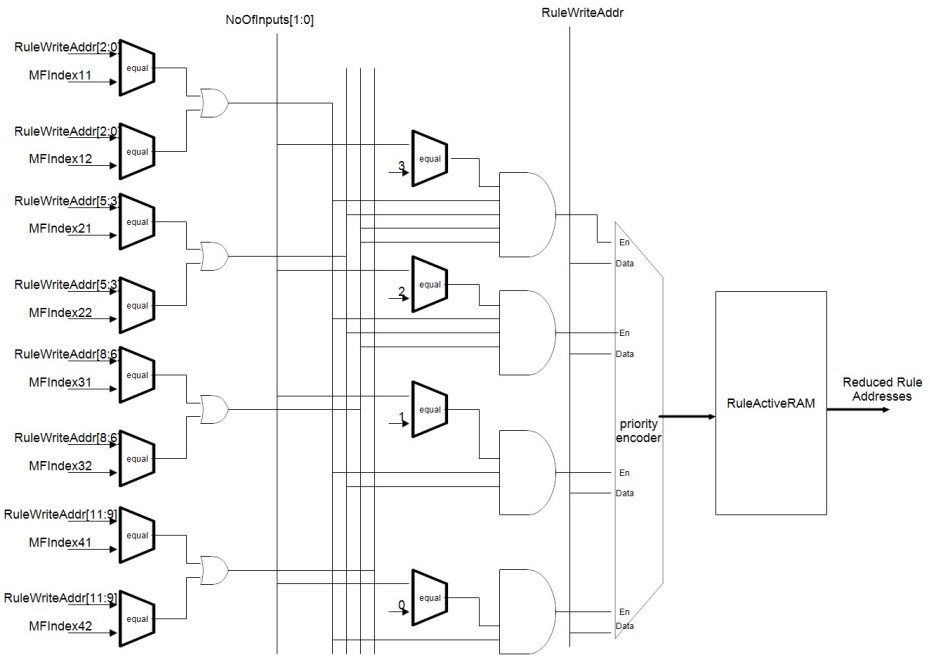


Figure 2.15: Circuit diagram of rule address generator for MRA2-OMF method

00000000	000 001 101 110 101 101 100 001 110 101 010 101 111 010 111 010 110 000 101
00000023	010 110 101 111 011 010 010 100 010 010 001 000 000 001 011 110 110 110 100
00000046	111 111 100 011 001 001 000 111 001 110 100 010 000 111 101 010 110 101 001
00000069	111 011 111 100 111 011 110 010 001 101 010 101 101 111 001 100 000 010 011
0000008c	101 110 100 111 001 100 000 000 101 111 110 000 100 101 001 100 110 010 101
000000af	110 000 110 011 111 001 010 001 111 001 110 000 101 101 001 001 101 000 011
000000d2	000 010 011 011 111 110 010 100 001 010 100 111 110 010 010 010 101 100 100
000000f5	000 101 011 010 110 101 100 000 001 110 001 011 000 011 110 011 010 100 011
00000118	111 100 111 111 001 001 011 001 001 001 010 010 101 011 100 010 110 000 111
0000013b	111 011 001 110 101 001 101 010 111 100 100 001 011 110 111 110 100 101 000
0000015e	101 000 101 010 000 101 010 100 110 100 000 111 010 100 010 100 000 010 011
00000181	111 010 001 000 011 011 100 011 100 001 001 111 100 111 101 000 010 001 001
000001a4	100 001 101 111 000 001 001 111 010 000 011 100 000 111 110 000 111 101 100
000001c7	111 001 101 110 000 101 000 100 001 001 111 110 001 101 000 000 010 111 001
000001ea	010 001 010 100 010 101 001 101 101 001 001 111 011 101 000 110 010 101 010
0000020d	001 001 011 100 101 100 001 010 101 101 110 010 000 000 101 111 000 000 101
00000230	101 100 010 100 000 011 001 110 000 110 000 010 100 001 011 000 111 000 101
00000253	010 010 011 010 100 001 011 011 000 010 001 111 011 101 000 101 111 011 011
00000276	100 100 111 001 011 101 001 111 111 000 011 110 010 010 001 111 110 001 101
00000299	000 101 110 110 011 011 011 111 111 100 010 110 100 011 011 100 100 101 001
000002bc	001 011 000 111 111 010 101 000 011 001 010 100 101 101 000 100 100 000 011
000002df	011 000 101 101 010 000 010 000 000 011 100 101 010 110 100 011 000 000 111
00000302	111 011 110 011 000 001 000 101 011 100 100 110 110 111 011 000 010 111 101
00000325	010 000 010 010 011 010 011 011 010 100 001 110 001 011 111 001 000 001 110
00000348	011 100 111 010 011 010 000 110 100 011 010 010 111 000 111 000 010 101 011
0000036b	001 001 010 101 111 101 000 011 010 111 110 000 000 010 010 100 010 010 110
0000038e	100 100 100 010 011 100 011 101 100 101 100 111 111 000 011 111 011 011 100
000003b1	100 111 001 001 100 010 101 111 101 000 011 001 100 000 001 110 010 111 001
000003d4	110 101 100 111 100 011 010 001 110 101 110 010 000 111 000 000 101 001 011
000003f7	001 001 001 111 110 101 111 011 001 110 010 100 001 011 010 000 000 001 101

Figure 2.16: Complete RuleBase Memory

but if the rule set only has 6 active rules, then the qualified rules to fire should be 6 and effectively reduce the total number of rules from 16 to 6 and further reduces the input-output time. Here the Rule Active RAM stores the reduced rules by comparing the Overlap Indexes with the effective rule address formed by each rule. The inference module processes these reduced rule addresses for its execution.

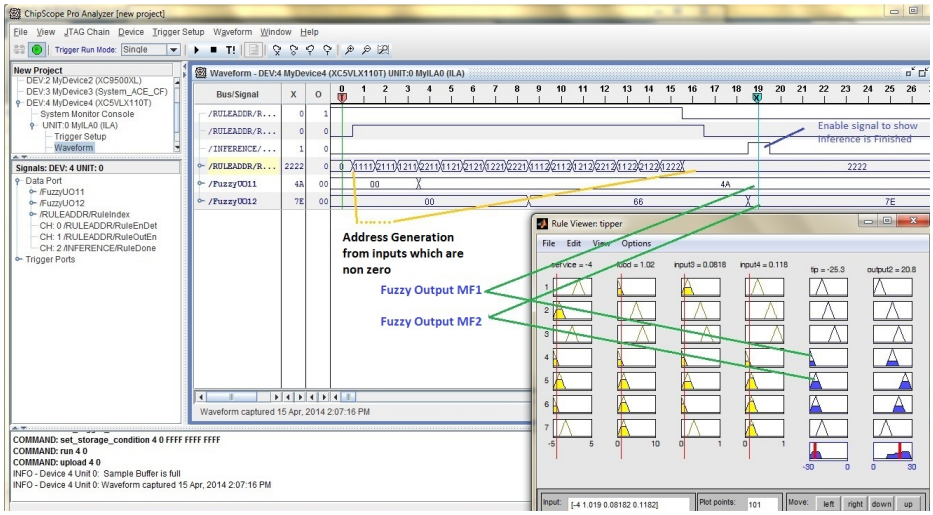
(a) 4 input MF Value (b) 3 input MF Value (c) 2 input MF Value

(d) 4 input MF Index (e) 3 input MF Index (f) 2 input MF Index

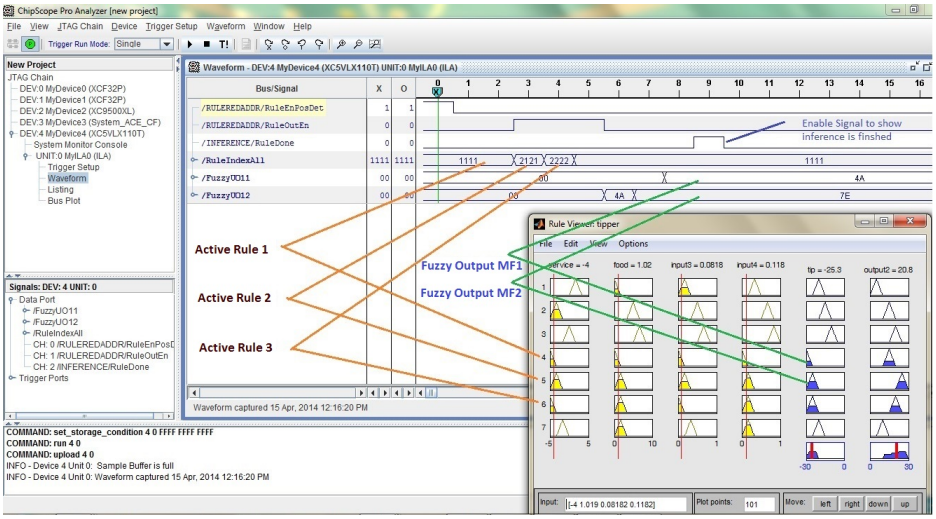
Figure 2.17: Dependency of 2-OMF Reduction on Rule Base Structure.

2.6 Simulation Results and Performance Evaluation

In this section, the performance of proposed MRA2-OMF rule reduction technique is compared with 2-OMF rule reduction method and complete rule base implementation using simulation results. Xilinx ISIM 14.5 simulator was used here to simulate all three architectures. A complete rule base set is presented in Figure 2.16, Where the blue lines



(a) Fuzzy Operation using 2-OMF Method



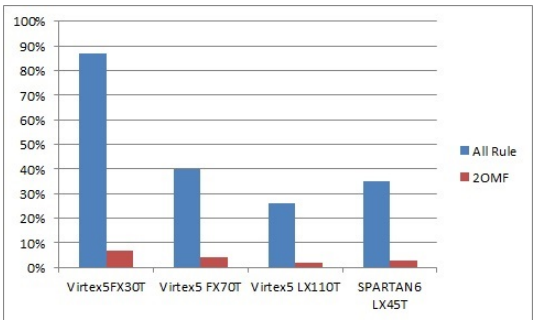
(b) Fuzzy Operation using MRA2-OMF Method

Figure 2.18: FLC operation on Virex5LX110T Board

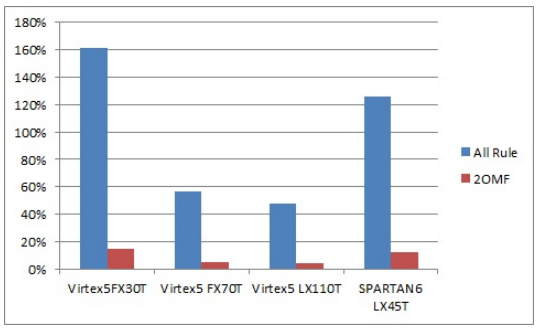
shows address of the rule base and 3 bit value shows the index of consequent. Figure 2.17 shows reduced rule membership values with its index numbers of 2-OMF rule sets.

Using the Chipscope pro Analyzer [118] the FLC operation with 2-OMF method is compared with the MRA2-OMF is presented in Figure 2.18. The Logic utilization comparison in Figure 2.19 shows the computational area has been reduced considerably by using the 2-OMF method and MRA2-OMF methods. The final implementation results are summarized in Table 2.3. From this, it is observed that the MRA2-OMF method provides superior implementation in terms of chip area and cycle time than 2-OMF method.

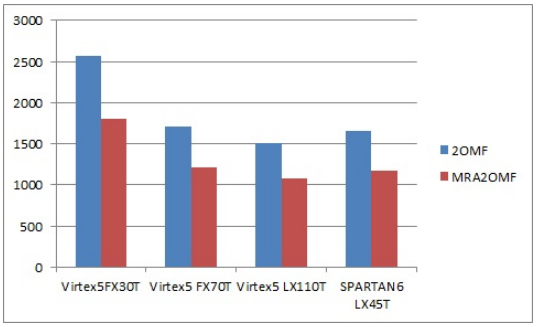
Thus with a 16-bit data path, the 2-OMF DFLC is capable of handling 7.2 to 4.15 MFLIPS considering rule range of [1:16] from (2.8) for n=4 and the MRA2-OMF DFLC is capable of handling 21.5 to 4.15 MFLIPS considering rule range of [1:16] from (2.10) for n=4. Here, the



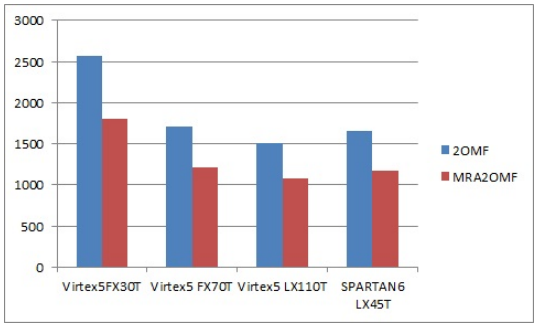
(a) Slice Registers for All Rule Versus 2-OMF method



(b) Slice LUTs for All Rule Versus 2-OMF method



(c) Slice Registers for 2-OMF method Versus MRA2-OMF method



(d) Slice LUTs for 2-OMF method Versus MRA2-OMF method numbers

Figure 2.19: Comparative Analysis of Logic Utilization.

Table 2.3: Implementation results for proposed methods

Proposed Methods	BRAM utilized		FFs	LUTs	Speed(MHz)	Power(μW)
	32K	18K				
2-OMF	3	**1**	1506	3368	145.658	27
MRA2-OMF	3	**1**	992	2327	147.923	26

increase in rules consumes more processing time and results less inference speed. Table 2.4 presents the comparative study of proposed two methods with the different architectures in literature. These fuzzy implementations on FPGAs are targeted for a specific application with fixed fuzzy parameters. Whereas the DFLC supports various fuzzy parameters as per the specification defined in section 2.2. The proposed DFLC outperforms all other implementations with its superior fuzzy inference speed.

2.7 Summary

The VLSI architecture of the FLC opens up a new line of approach to several explorations. Firstly, fuzzification, inference, and defuzzification with the total rule base was implemented and observed that it consumed more area for all industry based FPGA's. Secondly, the

Table 2.4: Comparison proposed methods with other FPGA Implementations

Year	Application	Inputs	Input MFs	Outputs	Output MFs	No Of Rules	Defuzzifier	Speed	Device
1995	Controller [96]	2(6 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	1.67MFLIPS	SC4008
1996	CAD Truck Control [97]	2(8 Bit)	5-5 (Triang)	1 (8bit)	5(Triang)	11	MOA	1.25MFLIPS	XC4006
2001	Car Parking [98]	2(8/10/12 bit)	5-7 (Trap/Triang)	1 (8/10/12 bit)	7(singlton)	35	WAM	333/277/222 KFLIPS	FLEX 10K
2006	C0-Processor [99]	2(8 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	2.5 MFLIPS	XC3S200E
2006	General [100]	2(6 Bit)	7-7 (Triang)	1 (6bit)	5(singlton)	9/49	MOM	2.85,0.92 MFLIPS	XC2S200E
2006	Climate Control [101]	4(12 Bit)	7-7 (Triang)	2(12bit)	7(Triang)	16	COA	77KFLIPS	A54SX32A (Actel)
2008	Bit serial Arthmatic [102]	2(6 Bit)	3-3 (Triang)	1 (8bit)	9(singlton)	9	COG	5.26MFLIPS	EP1S80B956C6
2010	Mobile Robots [103]	2(12 Bit)	9-9 (Trap/Triang)	1 (12bit)	9(singlton)	81	COG	1.2MFLIPS	Spartan 3E
2011	MPPT [93]	2(16 Bit)	5-5 (Triang)	1 (16bit)	9(singlton)	-	COA	-	EP2C35
*	Proposed (2-OMF)	1 to 4(8/16 Bit)	(1 to 7)-(1 to 7) (Triang)	1 to 2 (8/16bit)	1 to 7(Triang)	1 to 2401	COG	7.2 to 4.15 MFLIPS	Virtex 5 LX110T
*	Proposed (MRA2-OMF)	1 to 4(8/16 Bit)	(1 to 7)-(1 to 7) (Triang)	1 to 2 (8/16bit)	1 to 7(Triang)	1 to 2401	COG	21.5 to 4.15 MFLIPS	Virtex 5 LX110T

inference processing time of this FLC is optimized, and this fast FLC is relatively simple and provides good performance/price comparison.

The superior performance of this approach serves as a basis for further fuzzy applications, where the 2-OMF and MRA2-OMF based methods have been investigated for rule reduction. The next chapter focuses the further development of these methods, partial rule base support, and online tunability option in FLC.

Chapter 3

Tunable Digital Fuzzy Logic Controller with rule reductions and Special Case Rule Base Support

Preface

This chapter, an architectural change in the tuning of FLC parameters is introduced. Serial communication is used to configure parameters remotely in real time. This feature of reconfigurability enables an user to change fuzzy parameters in real-time, eliminating repeated hardware programming. Hardware-software co-design architecture for the proposed Digital Fuzzy Logic Controller (DFLC) is developed on Xilinx Virtex5LX110T FPGA and seamlessly integrated with a MATLAB based Graphical User Interface (GUI) for re-configurability. A Modified Rule Active 3-OMF (MRA3-OMF) and A Modified Rule Active 4-OMF (MRA4-OMF) are also supported to improve the Fuzzy Inference Engine (FIE) performance. A mechanism to support canonical fuzzy IF-THEN rules with special cases of the fuzzy rule base is also included in DFLC architecture. The MATLAB GUI acquires the fuzzy parameters from users and a Universal Asynchronous Receiver/Transmitter (UART) is dedicated to data communication between the hardware and the fuzzy toolbox. DFLC peripheral integration with Micro-Blaze (MB) Processor through Processor Logic Bus (PLB) is established for Intellectual Property (IP) core validation. Analysis of this design is carried out using simulation results. The performance of the proposed system is compared with the Fuzzy Toolbox of MATLAB. Analysis of this design is achieved by using Hardware-In-Loop (HIL) test to control various plant models in MATLAB/Simulink environment.

3.1 Introduction

Fuzzy inference is a decision-making process to control a plant effectively, where the control mechanism is based on fuzzy theory. For low-speed control applications, a software that runs on a conventional microprocessor can perform the task but for high-speed applications, the fuzzy inference process demands faster processing. It has been observed through software implementations in a conventional microprocessor, the inference speed is limited to 10K FLIPS (Fuzzy Logic Inferences Per Second) and is convenient for the majority of current consumer applications. These systems suffer from the drawback of slow speed and higher resource consumption. On the other hand, high-speed applications such as automotive control, space applications, and aircraft embedded control systems demand higher speed, which is time critical in nature. These applications are very complex in nature and use a significant amount of rules to express their complex behaviors. Considering the speed of operation they also require dedicated hardware that partially or fully implements the fuzzy logic controller. The previous chapter focused on rule reduction based hardware architectures for FLCs to reduce system latency and chip area. Here, the current chapter discusses system architecture of the FLC system, which provides remote re-configurability or can be tuned remotely according to the requirement of the application namely tunable FLCs. The parameters to program are mostly the FLC parameters, which need to tune the fuzzy logic algorithm running on hardware. FLC parameters, which are tuned include the number of inputs, the number of outputs, the number, and type of input membership functions, the number, and type of output membership functions and IF-ELSE rule base for fuzzy inference process, etc. This work selects FPGA as a targeted device for its advantage of re-configurability, parallel data processing capability, and the support of soft/hard core processors.

In section 2.5 an approach to implement a fast processing of search for a rule contributing to output using MRA2-OMF method was proposed and implemented. The limitation of this design was the assumption of the maximum overlapping or an uncertainty between Membership Functions (MFs) was limited to two. This was done considering most applications to work in this boundary. However, for complex systems and to achieve superior control for a variety of cases the system should support more than 3 Overlap Membership Functions (3-OMFs). In this work, rule active approach had been limited to Rule Active 3 Overlap Membership Function and Rule Active 4 Overlap Membership Function.

This Chapter introduces a Finite State Machine (FSM) to support canonical fuzzy IF-THEN rules with its special cases, support for "Partial Rule Base" and "Single Fuzzy Statements". The majority of the FPGA-based FLCs [119–121] have omitted this feature in their FLC implementations, but for a complete FLC inference mechanism these rules must be supported and tested.

3.2 Special Case Rule Base

As discussed earlier, In the FLCs the processing stage is based on a collection of logic rules in the form of IF-THEN statements as presented in (3.1), where the IF part is called the 'antecedent' and the THEN part is called the 'consequent'. The fuzzy rule base form the heart of fuzzy system with large number of rules.

$$R^l : IF\ X_1\ is\ M_1^l\ and\ X_2\ is\ M_2^l ... and\ X_n\ is\ M_n^l\ THEN\ Y\ is\ N_l \tag{3.1}$$

Where, $X_1, X_2, ...\ X_n$ are the input variables and Y is the output variable. M_n^l and N_l are linguistic values represented as fuzzy subsets of the respective universe of discourse U_i and V at input and output respectively. The rules in the form of (3.1) is called as canonical fuzzy IF-THEN rules, since they can include other types of fuzzy rules and fuzzy propositions as special cases, as presented in the following lemma.

Lemma 1. *The canonical fuzzy IF-THEN rules in the form of* (3.1) *include the following as special cases with m<n:*

a "Partial Rules":

$$IF\ X_1\ is\ M_1^l\ and ... and\ X_m\ is\ M_m^l\ THEN\ Y\ is\ N_l \tag{3.2}$$

b "OR Rules":

$$\begin{aligned} &IF\ X_1\ is\ M_1^l\ and ... and\ X_m\ is\ M_m^l\ OR \\ &IF\ X_{m+1}\ is\ M_{m+1}^l\ and ... and\ X_n\ is\ M_n^l\ THEN\ Y\ is\ N_l \end{aligned} \tag{3.3}$$

c "Single Fuzzy Statement":

$$Y\ is\ N_l \tag{3.4}$$

Proof. The partial rule in (3.2) is equivalent to

$$IF\ X_1\ is\ M_1^l\ and ... and\ X_m\ is\ M_m^l\ and\ X_{m+1}\ is\ I\ and\ X_n\ is\ I\ THEN\ Y\ is\ N_l \tag{3.5}$$

Where I is fuzzy set in R with M_I = 1 for all $x \in R$. The proceeding rule is in the form of (3.1)) and proves 'a'. Based on the nonrational meaning of the logic operator 'OR' the 'OR

Rules' in (3.3) is equivalent to

$$IF\ X_1\ is\ M_1^l\ and...and\ X_m\ is\ M_m^l\ THEN\ Y\ is\ N_l \tag{3.6}$$

$$IF\ X_{m+1}\ is\ M_{m+1}^l\ and...and\ X_n\ is\ M_n^l\ THEN\ Y\ is\ N_l \tag{3.7}$$

From (3.3) the two rules in (3.6) and (3.7) are special case of (3.1), this proves 'b'. The fuzzy statement is equivalent to

$$IF\ X_1\ is\ I\ and...and\ X_n\ is\ I\ THEN\ Y\ is\ N_l \tag{3.8}$$

Which is in the form of (3.1) and this proves 'c'. □

The current Chapter provides a State Machine (SM) to support these special cases of the rule base to implement each rule in hardware.

3.3 Modified 3-OMF Rule Active (MRA-3OMF) and Modified 4-OMF Rule Active (MRA-4OMF) Rule Reduction

The concept of 2-OMF has been widely used [117] in reducing computational time of fuzzy systems, especially in hardware development of FLC as it eradicates the nonlinear dependency between a number of inputs and computational complexity. However, this rule reduction method is limited by the assumption of maximum 2 overlaps. However, this stipulation of segregation of input space with a maximum of two overlapping membership functions causes major worry in an accuracy of the FLC, which is circumstantial in a majority of non-linear FLC system design. However, when it is employed to control a non-linear system, where the membership functions are overlapped with more than two over their input space, this system fails to provide expected accuracy. When the number of overlaps increases, the performance can be affected considerably.

Thus to keep the simplicity of the 2-OMF intact, while addressing its major disadvantage, 3-OMF and 4-OMF FLC is proposed here and incorporated in FLC hardware system design. The advantage of this design over the conventional counterpart can be understood considering a case, where fuzzy logic antecedent MFs are as shown in Figure 3.1. A system is considered where, $X1 \leq x \leq X2$. Any input in this region, is fuzzified to provide more than two non-zero values since there are more than two overlapping MFs. Further, it is observed that the number of non-zero values in a fuzzified input cannot exceed the number of overlapping MFs. If 2-OMF is applied on these set of input MFs, then useful data would

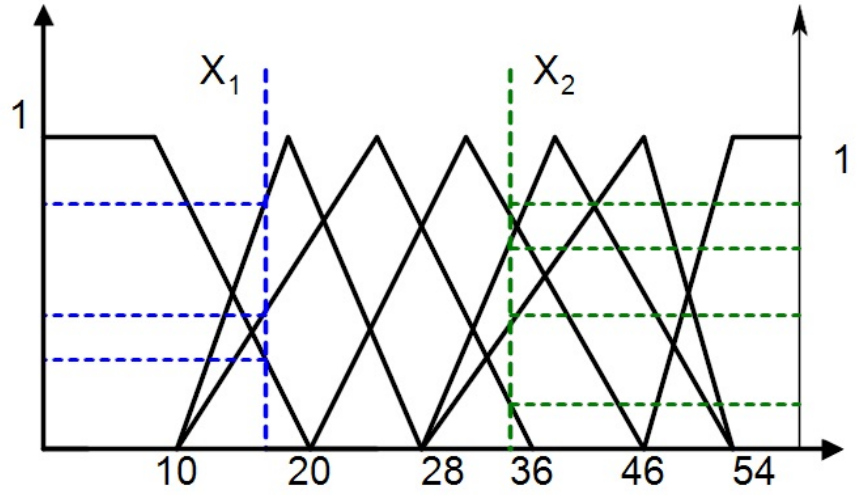


Figure 3.1: Less than 4 fuzzy membership functions overlapping at once

be lost. 2-OMF will only return two non-zero fuzzy values in place of three or four if input in the range of $X1$ to $X2$ input space. This results in non-accurate control output, which is proportional to the weight of the discarded membership function and its implication on the rule base. Support of 3 and 4 overlap methods in the design can allow the user to vary a number of overlaps in conjunction to the complexity of control system. Table 3.1 presents the rule search space with the number of overlaps. To limit the computational complexity in hardware, DFLC is modified to support up to four overlaps.

It is in general that from (2.6), for 3-OMF $N_i^{Over} = 3$ and 4-OMF $N_i^{Over} = 4$. The maximum number of rules derived is

$$N_{3-OMF} = \prod_{i=1}^{n} N_i^3 \tag{3.9}$$

$$N_{4-OMF} = \prod_{i=1}^{n} N_i^4 \tag{3.10}$$

Hence from (2.10) it can be seen that

$$N_{MRA3-OMF} = \prod_{i=1}^{n} N_i^3 - N_{null} \tag{3.11}$$

$$N_{MRA4-OMF} = \prod_{i=1}^{n} N_i^4 - N_{null} \tag{3.12}$$

The fraction F_{3A} of MRA3-OMF rules with 3-OMF rules and F_{4A} of MRA4-OMF rules with

4-OMF rules derived as

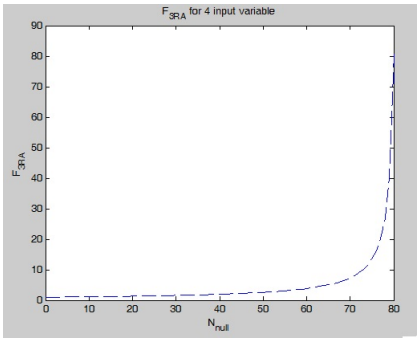
$$F_{3A} = \frac{\prod_{i=1}^{n} N_i^3}{\prod_{i=1}^{n} N_i^3 - N_{null}} \tag{3.13}$$

$$F_{4A} = \frac{\prod_{i=1}^{n} N_i^4}{\prod_{i=1}^{n} N_i^4 - N_{null}} \tag{3.14}$$

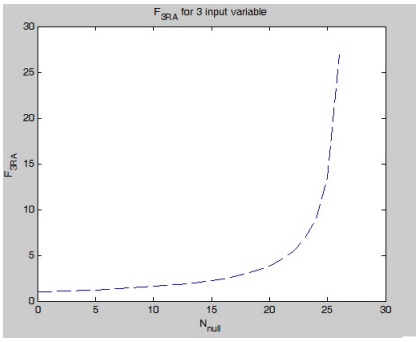
and the fraction F_{3RA} of MRA3-OMF rules with total rules and F_{4RA} of MRA4-OMF rules with total rules are

$$F_{3RA} = \frac{\prod_{i=1}^{n} N_i^{Over}}{\prod_{i=1}^{n} N_i^{Over} - N_{null}} \tag{3.15}$$

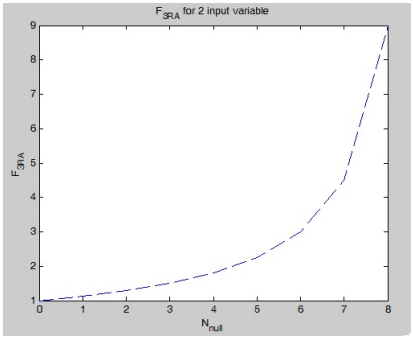
$$F_{4RA} = \frac{\prod_{i=1}^{n} N_i^{Over}}{\prod_{i=1}^{n} N_i^{Over} - N_{null}} \tag{3.16}$$



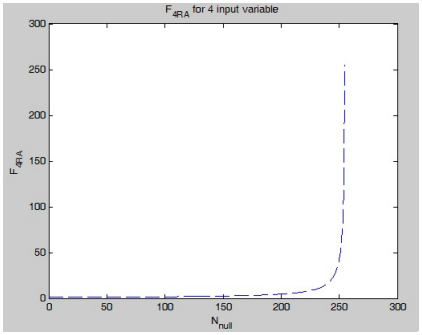
(a) F_{3RA} for 4 input variables



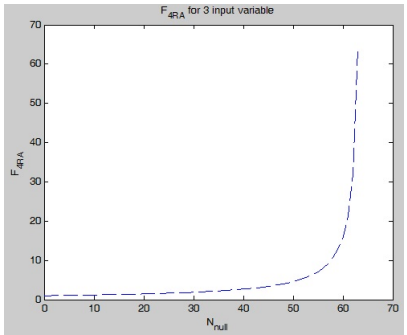
(b) F_{3RA} for 3 input variables



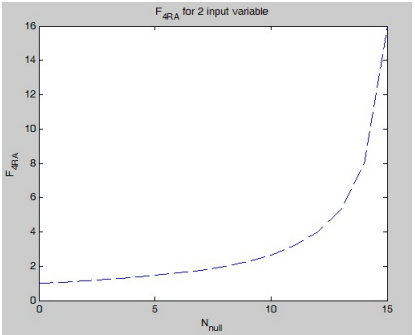
(c) F_{3RA} for 2 input variables



(d) F_{4RA} for 4 input variables



(e) F_{4RA} for 3 input variables



(f) F_{4RA} for 2 input variables

Figure 3.2: MRA3-OMF and MRA4-OMF fractional values with different input variables.

Figure 3.2 shows the plot between 3-OMF rules with MRA3-OMF rules and 4-OMF rules with MRA4-OMF rules, where Figure 3.2a, 3.2d, 3.2b, and Figure 3.2e, 3.2c, 3.2f are plotted for 4,3, and 2 inputs respectively.

Table 3.1: Rule search space for varying n and Overlaps

Overlaps	N_{cells} (n=1)	N_{cells} (n=2)	N_{cells} (n=3)	N_{cells} (n=4)
2	2	4	8	16
3	3	9	27	81
4	4	16	64	256
5	5	25	125	625
6	6	36	216	1296
7	7	49	343	2401

3.4 Configurable DFLC IP Core

In the present architecture, two systems have been proposed, one for fuzzy validation and another for IP validation. Any FPGA chip can be used in this work for hardware deployment. The software implementation of the fuzzifier, defuzzifier, and GUI for control data transfer has been completed on the MATLAB platform as System Method 1 (SM1) and MicroBlaze processor as a System Method 2 (SM2). SM1 can be used for fuzzy validation to compare with existing fuzzy toolbox, and SM2 can be used to validate IP Core by producing different test vectors. UART serial communication port and Processor Local Bus is used for 2-way data transfer between software and hardware for SM1 and SM2 simultaneously. The tunability option is provided in the IP core architecture. The term tunability here is refined to the point that architecture provides re-configurability of parameters. The specifications for the IP core are as follows.

- Number of Inputs: 4 (Maximum, Configurable)
- Number of Outputs: 2 (Maximum, Configurable)
- Shape of Membership Function: Triangular
- Number of MF for each input and output: 7 (Maximum, Configurable)
- MF Overlapping Degree: 2, 3 or 4 (Configurable)
- Implication model: Mamdani

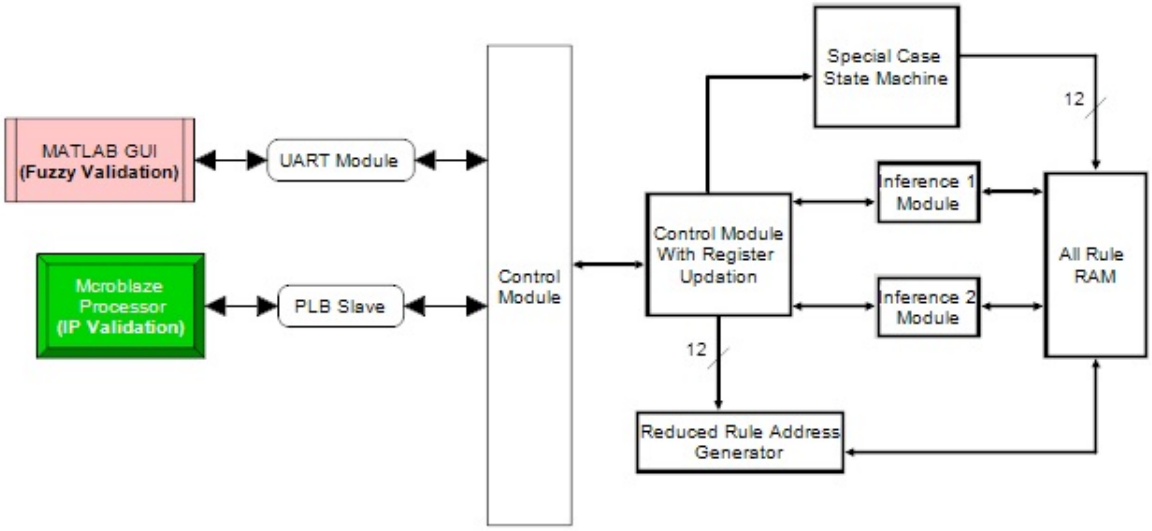


Figure 3.3: System Model for IP and Fuzzy Validation of Inference IP Core

- Aggregation model: Mamdani
- Inference rules: Field programmable
- The system can be configurable in the field through a GUI application running on a MATLAB.

The fuzzy memory map is designed and provided for software interaction with the fuzzy inference IP core. Here memory map refers to process of sending the fuzzy parameters like the complete fuzzy rule base, number of input MFs, a number of output MFs, a number of inputs and outputs, and control signals.

Figure 3.3 presents the system model to test the proposed architecture for fuzzy inferences with the existing MATLAB fuzzy toolbox. The control module extracts the memory map of Figure 3.4 into its internal registers and generates the control signal to start the inference process. This memory map supports both for 8-bit and 16-bit widths based on the accuracy of the system which is user selectable. Two inference modules of the MAX-MIN circuit are used to support two outputs of the IP core. From the minimum configuration to a maximum configuration, the memory map varies from the size 17 Bytes to 7254 (Approx. 7K) Bytes. These values are calculated by taking one input, 1 rule, and one output parameter values for minimum configuration and four inputs, 2401 rules, and two outputs parameter values for maximum configuration. From Figure 3.4 it can be seen that the 00H address location provide the control signals RE (Rule Enable) and IE (Input Enable). From Table 3.2, the process of the Inference will start its operation by configuring

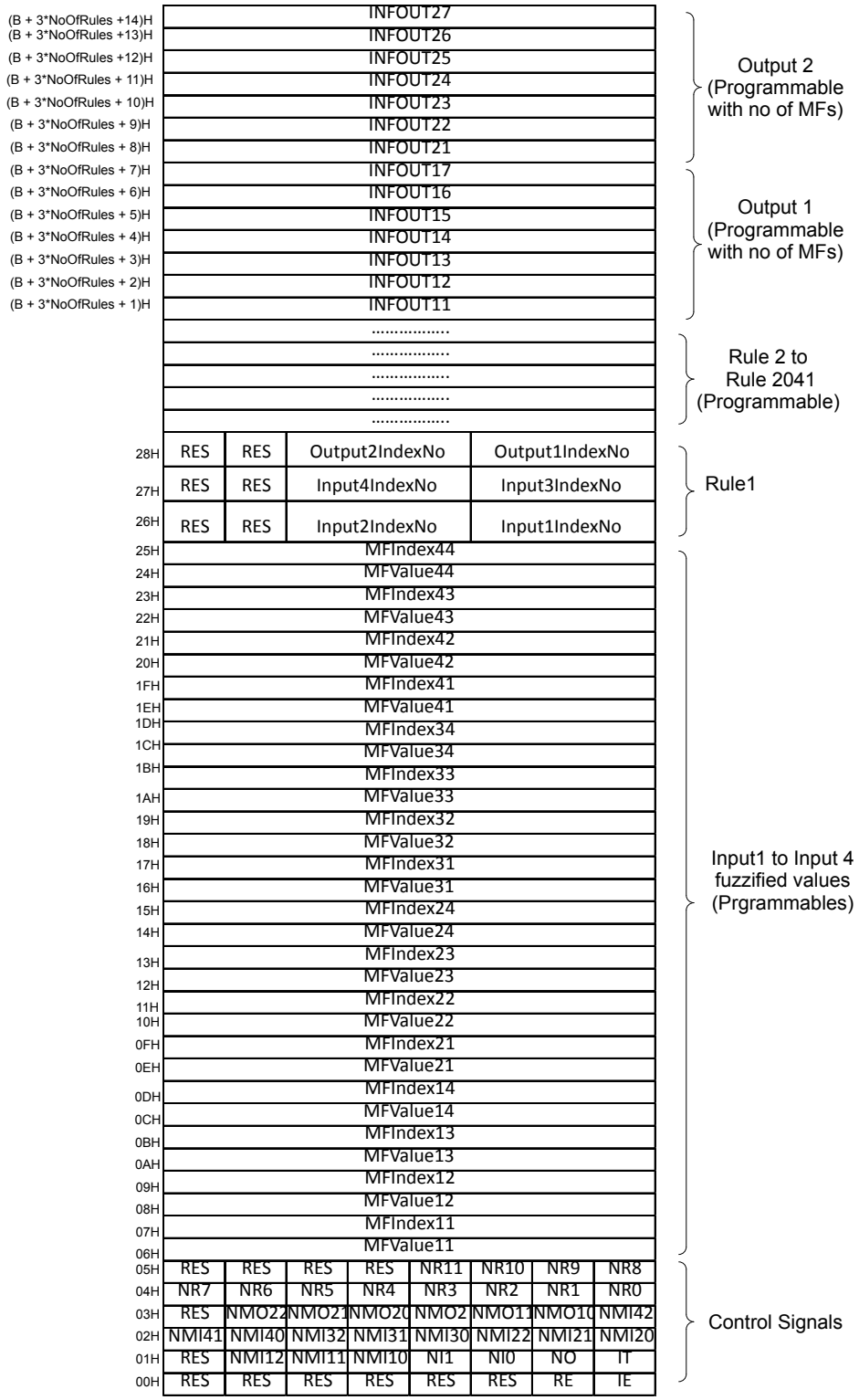
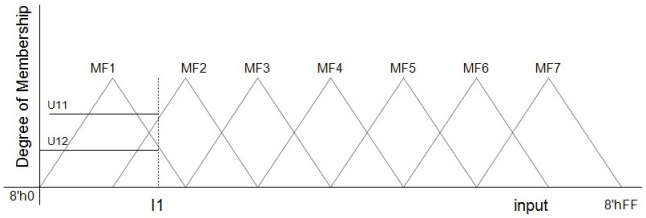
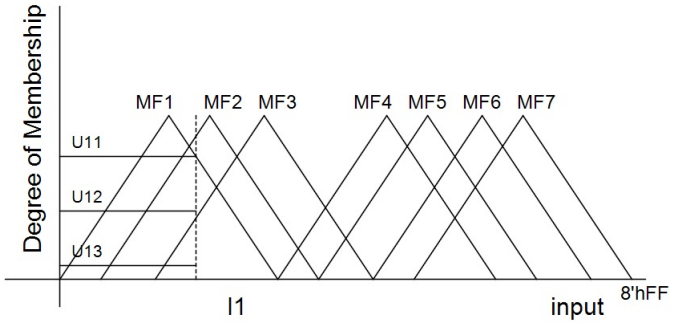


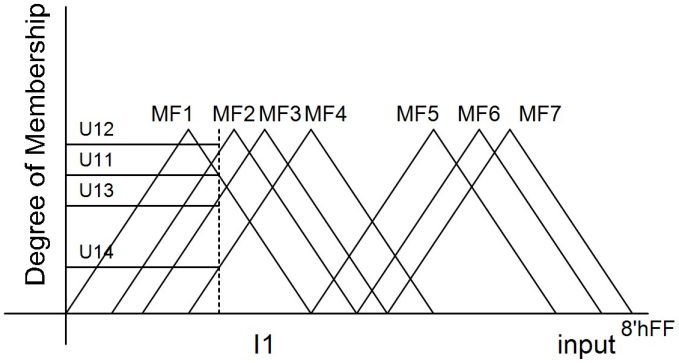
Figure 3.4: Memory Map to support different user configurations according to the system specification



(a) Two Overlap membership functions values



(b) Three Overlap membership functions



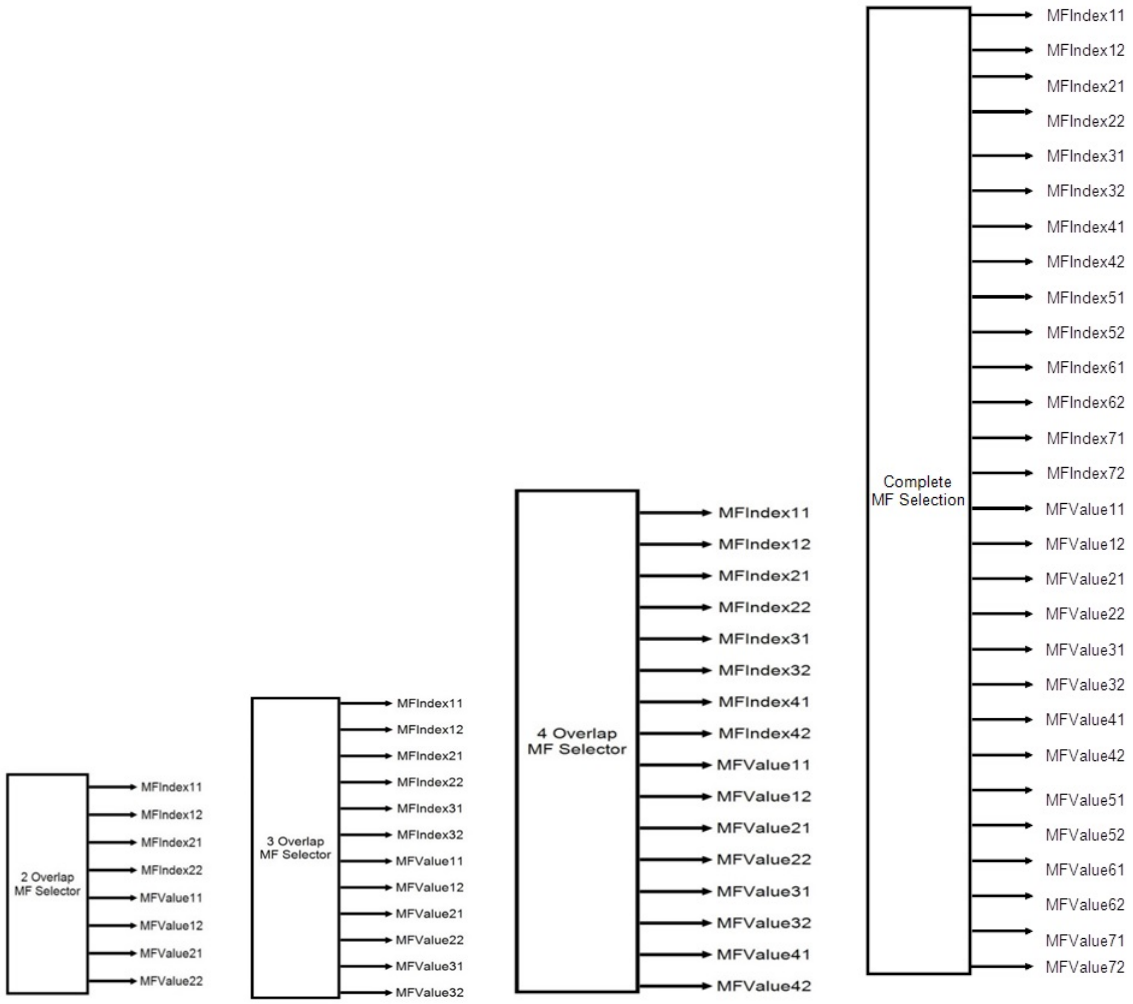
(c) Four Overlap membership functions

Figure 3.5: Rule reduction methods supported according to the present architecture

RE and IE.

Table 3.2: Control Signal Description to start different FLC programming options

Control Signals	Values	Description
{RE, IE}	00	No Inference Operation
{RE, IE}	01	With existing Rule Base and fuzzy parameters start inference for new input
{RE, IE}	10	Update new rule base, but there is no inference to start as if there is no new input
{RE, IE}	11	Update new rule base and start inference for new input available with new fuzzy parameters



(a) 2-OMF (b) 3-OMF (c) 4-OMF (d) All Rule

Figure 3.6: The rule-sector outputs for 2-OMFs, 3-OMFs and 4-OMFs rule reduction methods and all rules.

From Figure 3.4, for a 8-bit depth memory map the 01H to (B + 3 * No of Rules + 145)H address location (here 'B' is the variable which ranges from 08H to 25H for different configurations) is set for fuzzy parameters where

a. The parameter NO (Number of Outputs) is a single bit value to support maximum configurable outputs as two. Value '0' for a single output and '1' for two outputs.

b. The parameter NI (Number of Inputs) is a 2-bit value to support maximum 4 configurable inputs. Value '00' is issued for single input, '01' for two inputs, '10' for three inputs and '11' for four inputs.

c. The parameter NMI1, NMI2, NMI3, and NMI4 (Number of Membership Functions) relates to a number of membership functions at input for input 1,input 2, input 3, and

input 4 respectively. NMO1 and NMO2 relates to number of membership functions at output for output 1 and output 2 respectively. These parameters are assigned 3-bit value to give maximum configurable MFs of 7 to support full four inputs two output system. Value '000' for single MF, '001' for 2 MFs, '010' for 3 MFs, '011' for 4 MFs, '100' for 5 MFs, '101' for 6 MFs, and '110' for 7 MFs are used.

d. The parameter NR (Number of Rules) is a 12-bit value to support maximum 2401 configurable rules.

e. The parameter MFValueXY is a non-zero fuzzified value of a membership function where X denotes the input number, and Y denotes membership index number.

f. The parameter MFIndexXY is an MFValueXY's index number. The index value varies from '000' to '110' to support maximum configurable seven membership functions.

g. Each rule consumes 3 bytes with the corresponding index numbers of consequents and antecedents in the fuzzy rule base.

h. Modus ponens reasoning method for a mamdani's fuzzy inference strategy gives seven fuzzy set values on outputs with parameter INFOUT1Y and INFOUT2Y. Here Y denotes output membership index number.

Three methods of searching for the rule, which contributes inference output are shown in Figure 3.5. The rule selector module for 2-OMFs, 3-OMFs, and 4-OMFs with respect to All Rule is presented in Figure 3.6. Figure 2.8 illustrates the condition of a two input one output system where it fires at most two fuzzy sets for overlapping factor 2 and results not more than 4 rules. In the same way for the 3-OMFs case inference module has at most 2^3 = 8 and for 4-OMF case the inference module has 2^4 = 16 rules. The modified Rule Active method discussed in section 3.3 is applied in the architecture. This reduced the inference process time further.

3.4.1 State Machine for Partial and Complete Rule Generation

In the DFLC architecture described here, the mealy FSM is proposed and acts as a control unit. The main objectives of special state machine are to initiate rule base processing, generate special case rules like partial and single fuzzy statements. This state machine has 19 states, Where 3 states are used for control purpose and 16 states are used to support different

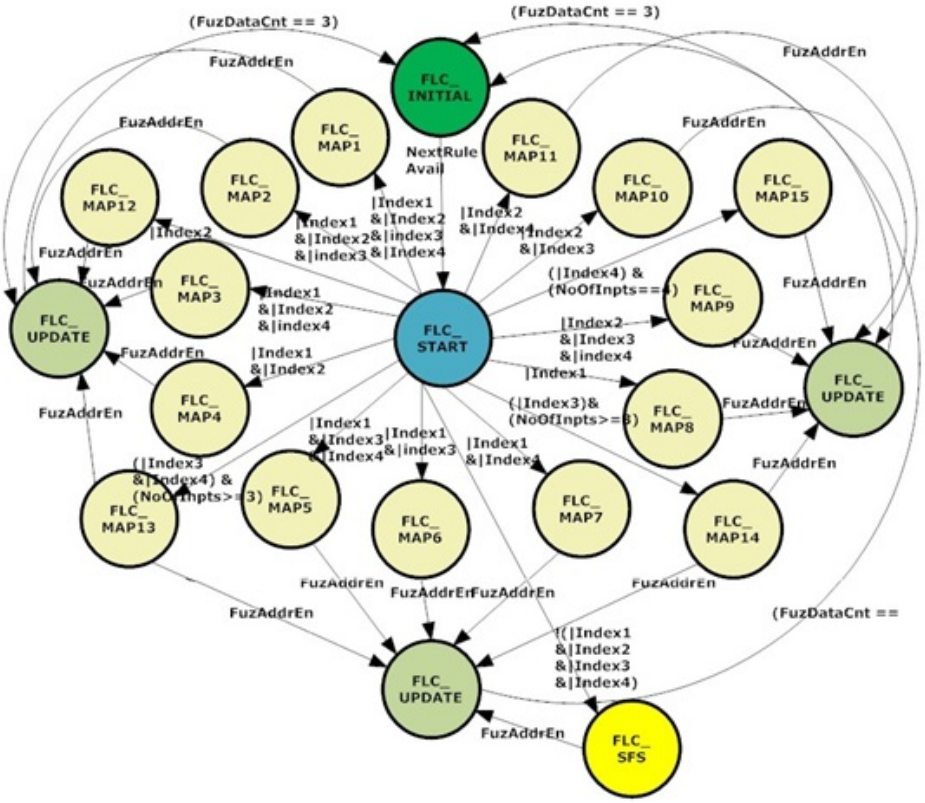


Figure 3.7: Finite State Machine to support special rule cases

special case rules. Figure 3.7 presents the controller state diagram. The state machine starts executing once the new rule is available in rule base memory. The state machine generates the address and places the corresponding output membership function in the 'All Rule memory'. The 'All Rule Memory' has an address depth of covering 2041 locations and the data width of 3 bits to store 2041 (7^4 Max) rules and to map seven membership functions at an output. For 4 (maximum configurable) input problem the rule base should support all special cases mentioned in section 3.2, for that the design considered the index numbers of each of its inputs and mapped them to the all rule memory by using a finite state machine. The address generation for each rule is obtained in each state are explained in Table 3.3. For partial rules, the missing input is mapped to all its combinations. Single Fuzzy Statement (SFS) special case is supported in state FLC_SFS. The main objective of this stat

3.4.2 Interfacing DFLC IP Core

To test the DFLC module with its counterpart MATLAB fuzzy toolbox, an interface is established using UART module. The complete DFLC Design module with its interfacing module is presented in Figure 3.8. The UART to Fuzzy IP Core is a simple communication

Table 3.3: State Transition Table

Present State	Next State	Transition Signal	Description
FLC_INITIAL	FLC_INITIAL	NextRuleAvail	Reset state which waits for new rule every time to Map the all rule address.
FLC_START	FLC_MAP1	|Index1 &|Index2 &|Index3&|index4	It's a complete rule where all inputs are active. In this state the write address is Index4, Index3, Index2, Index1
FLC_START	FLC_MAP2	|Index1 &|Index2 &|Index3	It's a partial rule where input 4 is absent. In this state the write address is Index4, Index3, Index2, Index1 where Index4<=Index4 +1 for 7(1 to 7) iterations
FLC_START	FLC_MAP3	|Index1 &|Index2 &|Index4	It's a partial rule where input 3 is absent. In this state the write address is Index4, Index3, Index2, Index1 where Index3<=Index3 +1 for 7 (1 to 7) iterations
FLC_START	FLC_MAP4	|Index&|Index2	It's a partial rule where input 3,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index3<=Index3 +1 for 7 (1 to 7) iterations then Index4<=Index4 +1 for 7 (1 to 7) to map all 3,4 locations.
FLC_START	FLC_MAP5	|Index&|Index3 &|index4	It's a partial rule where input 2 is absent. In this state the write address is Index4, Index3, Index2, Index1 where Index2<=Index2 +1 for 7 (1 to 7) iterations
FLC_START	FLC_MAP6	|Index1 &|Index3	It's a partial rule where input 2,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index2<=Index2 +1 for 7 (1 to 7) iterations then Index4<=Index4 +1 for 7 (1 to 7) to map all 2,4 locations.
FLC_START	FLC_MAP7	|Index1 &|index4	It's a partial rule where input 2,3 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index2<=Index2 +1 for 7 (1 to 7) iterations then Index3<=Index3 +1 for 7 (1 to 7) to map all 2,3 locations.
FLC_START	FLC_MAP8	|Index1	It's a partial rule where input 2,3,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index2<=Index2 +1 for 7 (1 to 7) iterations then Index3<=Index3 +1 for 7 (1 to 7), Index4<=Index4 +1 for 7(1 to 7) to map all 2,3,4 locations.
FLC_START	FLC_MAP9	|Index2 &|Index3&|index4	It's a partial rule where input 1 is absent. In this stae the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations.
FLC_START	FLC_MAP10	|Index2 &|Index3	It's a partial rule where input 1,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index4<=Index4 +1 for 7 (1 to 7) to map all 1,4 locations.
FLC_START	FLC_MAP11	|Index2 &|Index4	It's a partial rule where input 1,3 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index3<=Index3 +1 for 7 (1 to 7) to map all 1,3 locations.
FLC_START	FLC_MAP12	|Index2	It's a partial rule where input 1,3,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index3<=Index3 +1 for 7 (1 to 7), Index4<=Index4 +1 for 7 (1 to 7) to map all 1,3,4 locations.
FLC_START	FLC_MAP13	|Index3&|Index4 &(NoOfInputs>=3)	It's a partial rule where input 1,2 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index2<=Index2 +1 for 7 (1 to 7) to map all 3,4 locations.
FLC_START	FLC_MAP14	|Index3& (NoOfInputs>=3)	It's a partial rule where input 1,2,4 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index2<=Index2 +1 for 7 (1 to 7), Index4<=Index4 +1 for 7 (1 to 7) to map all 1,2,4 locations.
FLC_START	FLC_MAP15	|Index4 &(NoOfInputs==4)	It's a partial rule where input 1,2,3 are absent. In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7(1 to 7) iterations then Index2<=Index2 +1 for 7 (1 to 7), Index3<=Index3 +1 for 7 (1 to 7) to map all 1,2,3 locations.
FLC_START	FLC_SFS	!(|Index1 &|Inex2 &|Index3&|Index4)	Its single fuzzy statement where all inputs are absent and consequent is same for all rules In this state the write address is Index4, Index3, Index2, Index1 where Index1<=Index1 +1 for 7 (1 to 7) iterations then Index2<=Index2 +1 for 7 (1 to 7), Index3<=Index3 +1 for 7 (1 to 7), Index4<=Index4 +1 for 7 (1 to 7) to map all 1,2,3 locations.
FLC_MAP1 to FLC_MAP15 and FLC_SFS	FLC_UPDATE	FuzAddrEn	Once the consequent is written into corresponding Consequent memories (in this case 2 for maximum 2 output problems) this signal enables.
FLC_UPDATE	FLC_INITIAL	FuzDataCnt==3	This state makes Fuzzy Address to increment for 3 more locations to read next rule.

parser that can be used to access an internal bus via UART interface. The internal bus designed with address bus of 12 bits (to support 2401 rules) and data bus of 8 bits/16 bits (Option given in the core) based on the application. The UART module block diagram has

UART transmit and receive blocks, which share a common baud generator. The baud rate is set using two constants defined in the UART top core module, which is calculated as follows:

$$Q_{BAUDFREQ} = 16\frac{BaudRate}{GCD(GlobalClockFrequency * BaudRate)} \tag{3.17}$$

$$Q_{BAUDLIMIT} = 16\frac{GlobalClockFrequency}{GCD(GlobalClockFrequency * BaudRate)} - Q_{BAUDFREQ} \tag{3.18}$$

The baud rate generator parameters for baud rate 9600 bps and global lock frequency of 100MHz are configured in Verilog HDL as follows.

$$`define \quad Q\_BAUD\_FREQ \quad 12`h18 \tag{3.19}$$

$$`define \quad Q\_BAUD\_LIMIT \quad 16`h3CF1 \tag{3.20}$$

These two values are further used as an input signal to the UART top modules as shown in Figure 3.8.

3.4.2.1 MATLAB GUI and Operation

The MATLAB GUI to initiate the FPGA process is shown in Figure 3.9. The control register values, input fuzzified values with indexes, and the rule base are forced here as shown in Figure 3.10. For the purpose of testing, it can be chosen the same rule base formulated in the fuzzy toolbox. The inference output from DFLC is then read from COM1 port using the fread function of MATLAB and passed to defuzzification unit, which computes COG (2.4). The results from DFLC implemented in FPGA and fuzzy toolbox are then compared and displayed in GUI.

3.4.2.2 DFLC IP Core Peripheral Connection to MicroBlaze Processor

The MicroBlaze is a soft processor core designed for Xilinx FPGAs from Xilinx. As a soft-core processor, MicroBlaze is implemented entirely in the general-purpose memory and the logic fabric of Xilinx FPGAs. MicroBlaze's primary I/O bus, the CoreConnect PLB bus, is a traditional system memory mapped transaction bus with master/slave capability. The majority of vendor-supplied and third-party IP interface to PLB directly (or through a PLB to OPB bus bridge). Xilinx's EDK (Embedded Development Kit) is the development package for building MicroBlaze (and PowerPC) embedded processor systems in Xilinx FPGAs.

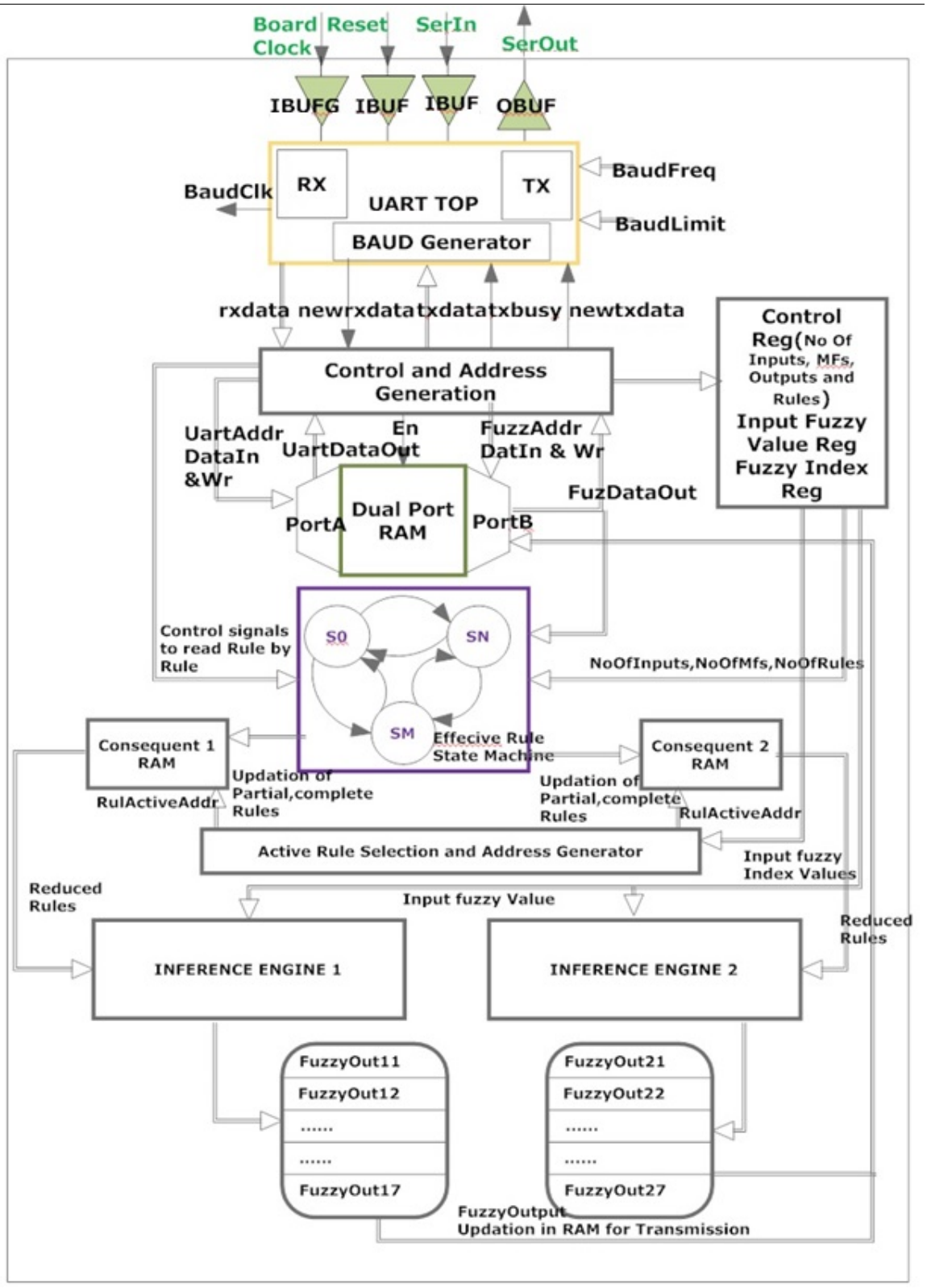


Figure 3.8: Detailed Design block diagram of DFLC

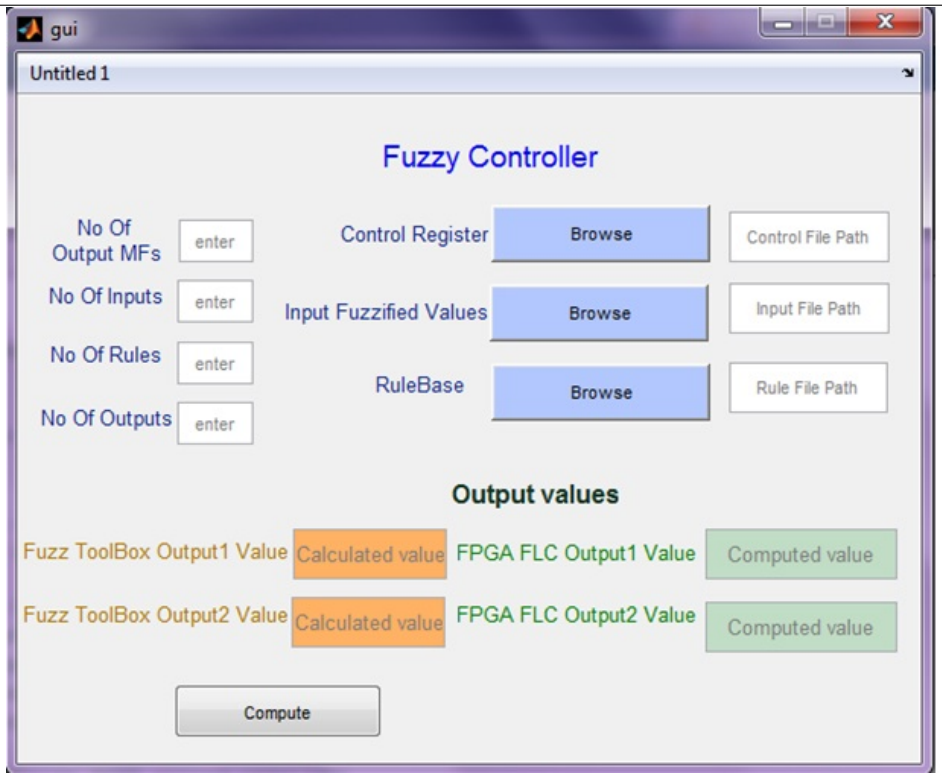


Figure 3.9: GUI to Initiate and Compare DFLC with Fuzzy tool box

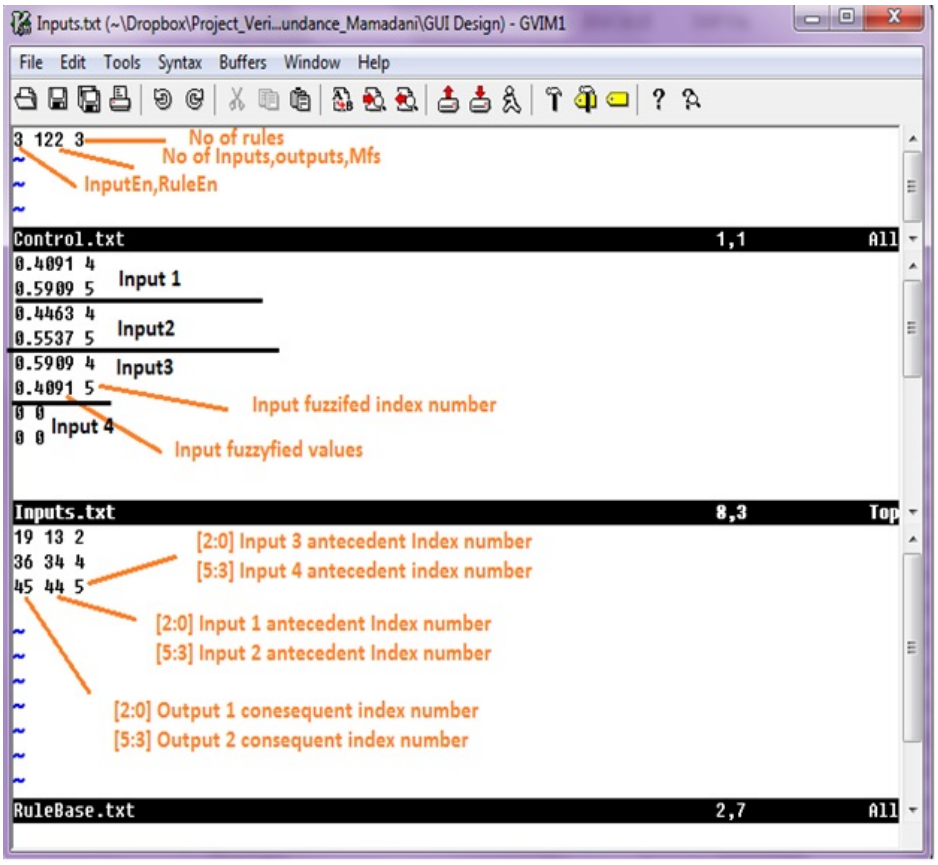


Figure 3.10: Configuration Register files of DFLC

Hosted in the Eclipse IDE, the project manager consists of two separate environments: XPS (Xilinx Platform Studio) and SDK (Software Development Kit). The work used XPS to configure and build the DFLC hardware specification. The XPS converts the DFLC specification into a User Peripheral and writes a set of scripts to automate the implementation of DFLC peripheral. The Xilinx SDK handles the software that executes on the DFLC peripheral. Figure 3.11 presents the interface connection of DFLC IP Core with PLB interface. Figure 3.12 illustrates the DFLC peripheral integration with MicroBlaze Processor.

3.5 Simulation Results and Analysis

In this section, the proposed MRA-3OMF, MRA4-OMF, and special case rule base supported DFLC performance has been compared to MATLAB Fuzzy Logic Toolbox for performance analysis. Generality of a DFLC allows it to work with various FIS (Fuzzy Inference System) structure files. A MATLAB GUI to an FIS structure file is used here for the purpose of comparison. The GUI extracts the parameters in control register file for a specific test case and generates testing models. The test model of special case rule "Partial Rule" is presented in Figure 3.13. The simulation result of "Partial Rule" test case is given in Figure 3.14, where the "Ready" signal validates the fuzzy output. The GUI displays the fuzzified output comparison between MATLAB Fuzzy Logic Toolbox and proposed DFLC.

The "Special Fuzzy Statement" special case rule simulation waveform is presented at Figure 3.15. From the waveform it can be inferred that the proposed system supports all rule structures in hardware implementation. Figure 3.17 shows the presented implementation is working under repeated rule update. In other words, the DFLC supports even if two rules have same antecedents.

Finally, the DFLC peripheral connection to the MicroBlaze processor goes through the PLB interface, with its register update as mentioned in Figure 3.4 and the performance is verified in Figure 3.18. The tuning of FLC parameters and the support of special rule case made the design more dynamic, but these extra features added more logic utilization to FPGA and it is observed in Table 3.4 by comparing with our previous proposed systems. A cycle time of value 6.608 ns was derived from the maximum clock frequency of the design using Virtex 5 LX110T FPGA. The latency of 46.256 ns is calculated for a 4 input Test vector with 3 active rules.

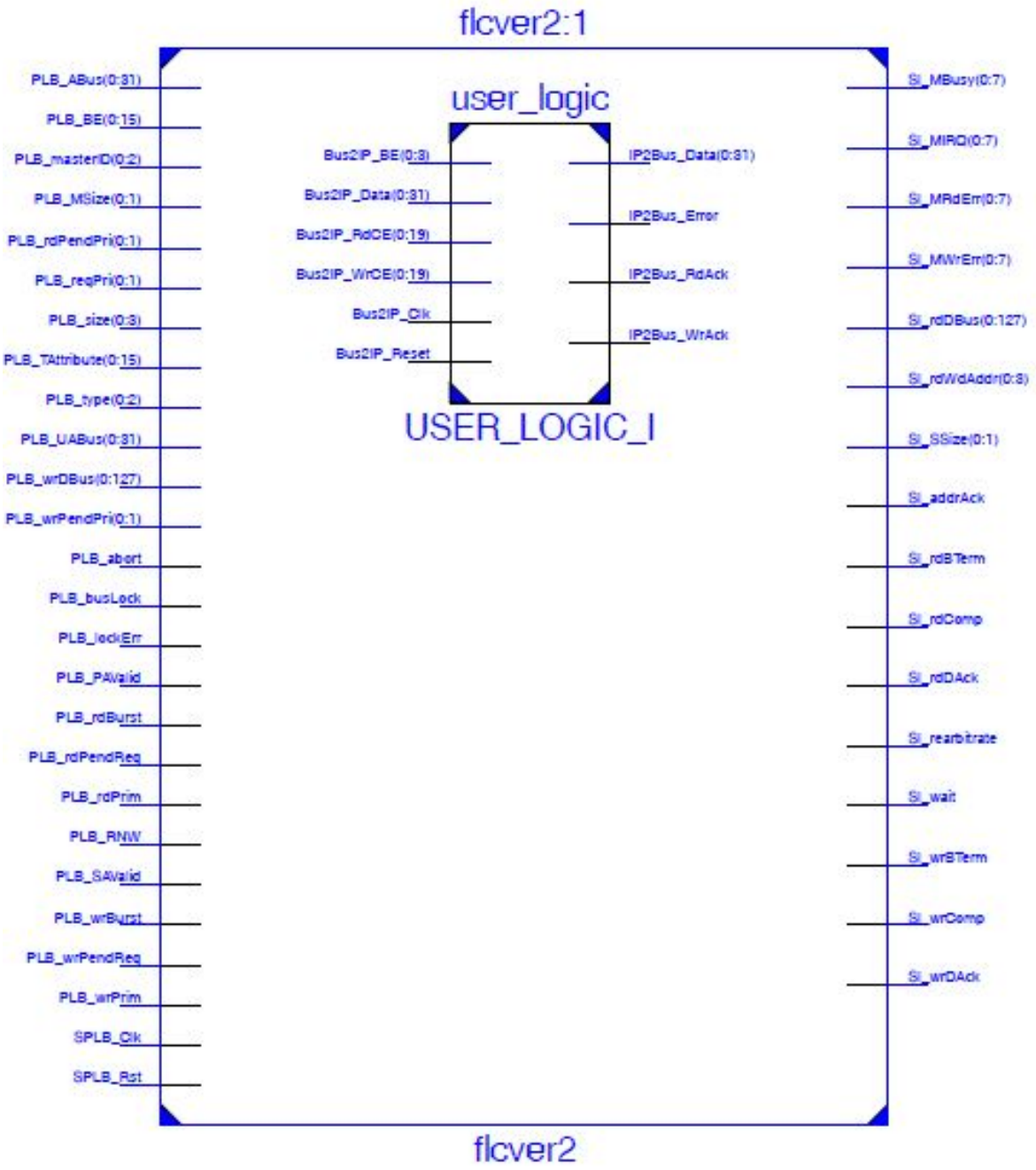


Figure 3.11: DFLC Peripheral Connection PLB Interface to Microblaze Processor

Table 3.4: Hardware implementation: Comparison of all proposed methods

Proposed Methods	BRAM utilized		FFs	LUTs	Cycle Time	Latency
	32K	18K				
2-OMF	0	1	1506	3368	6.865 ns	137.3 ns
MRA2-OMF	0	1	992	2327	6.76 ns	47.32 ns
Tunable MRA2-OMF-SpecialRule	0	1	2056	4126	6.608 ns	46.256 ns

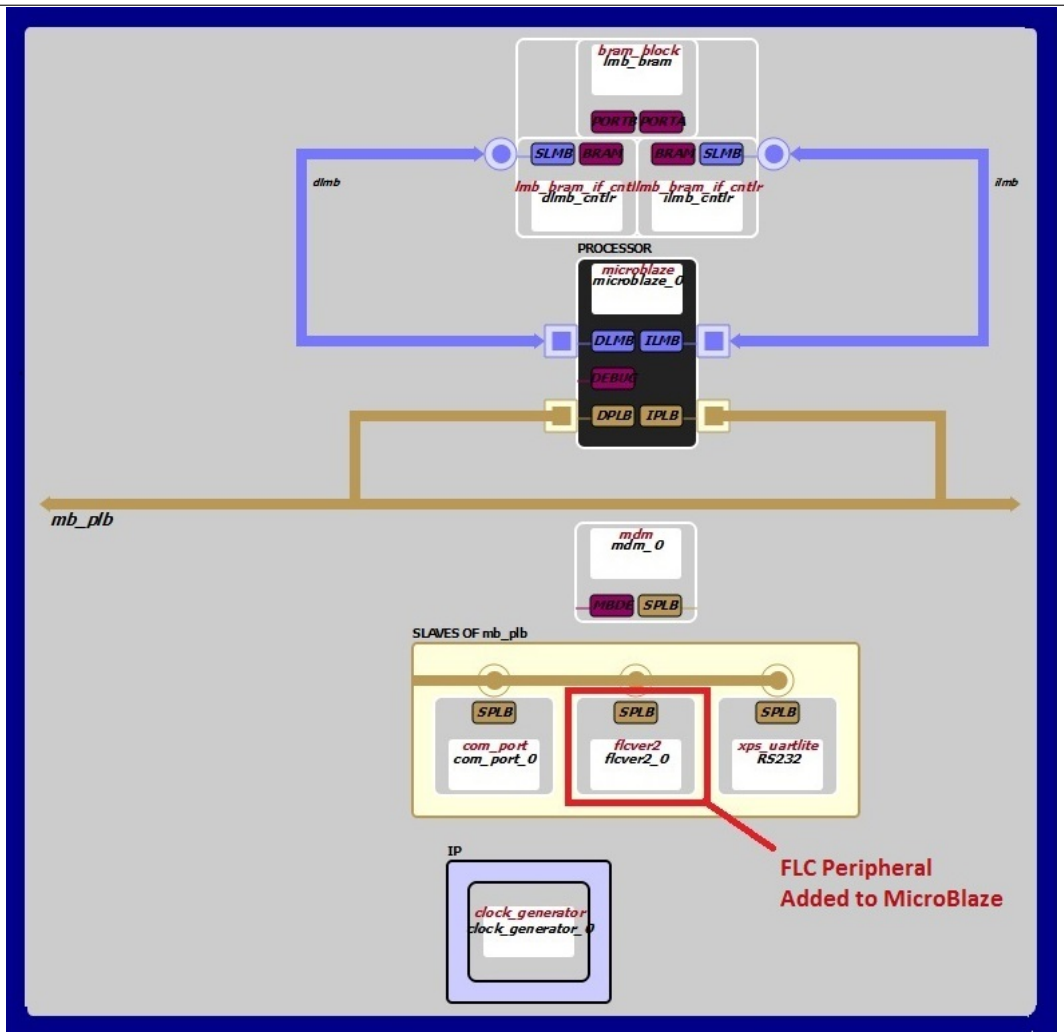


Figure 3.12: DFLC Peripheral to Microblaze Processor

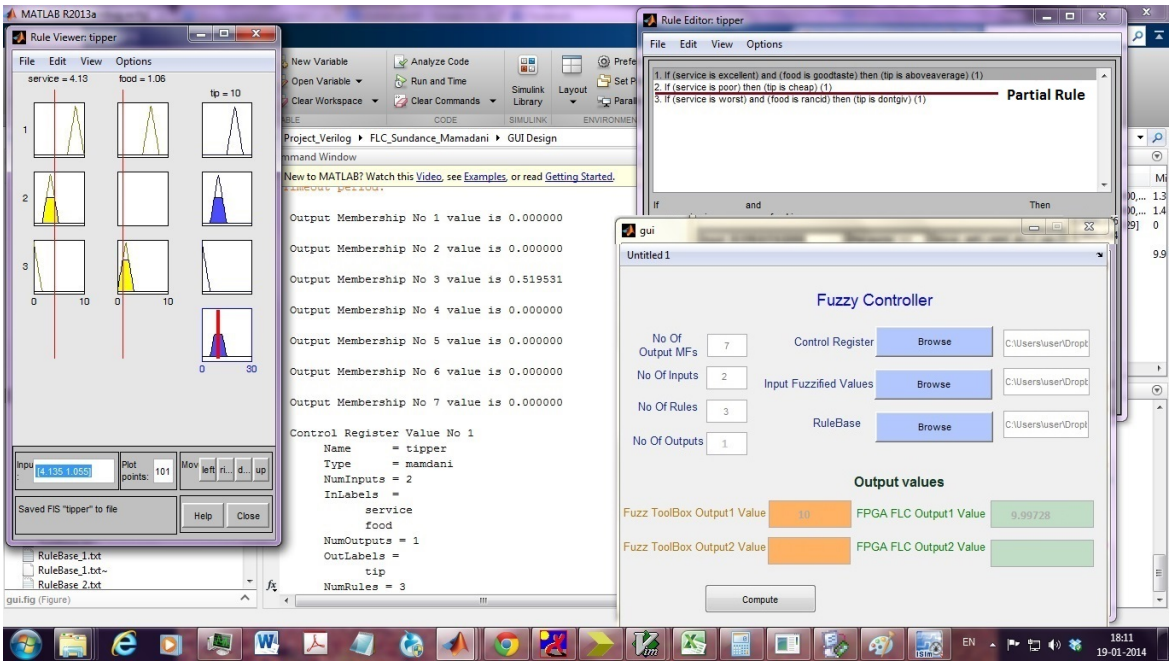


Figure 3.13: Test Model for partial rule support

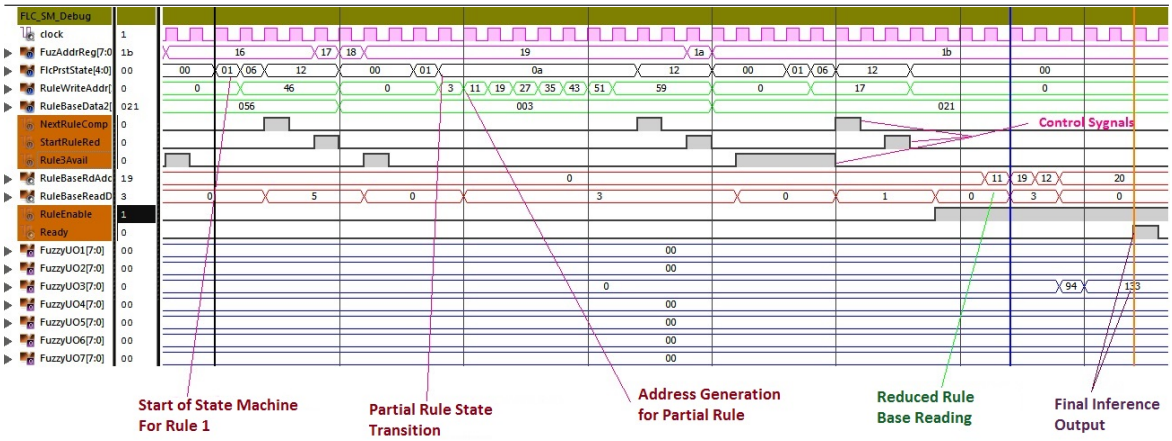


Figure 3.14: Simulation waveform of partial rule support

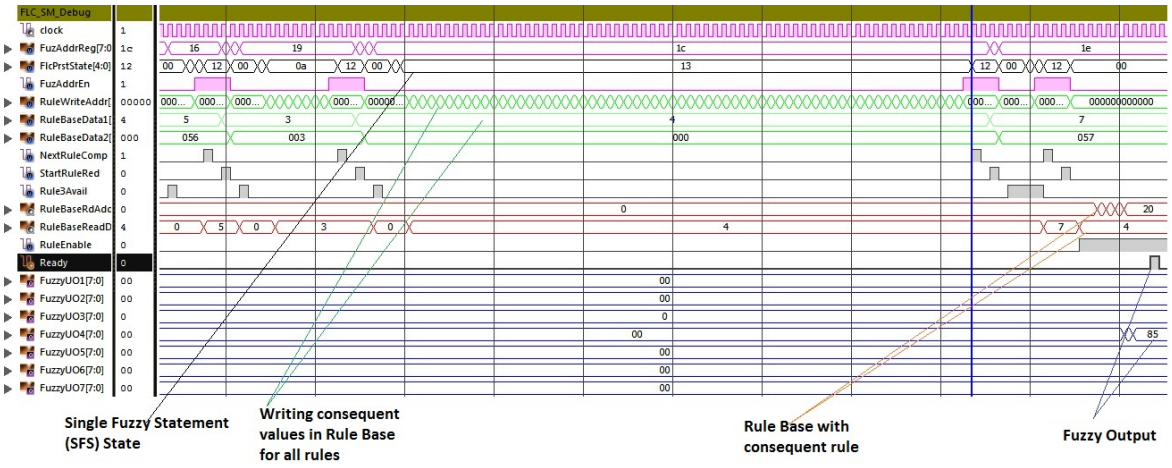


Figure 3.15: Simulation waveform of single fuzzy statement

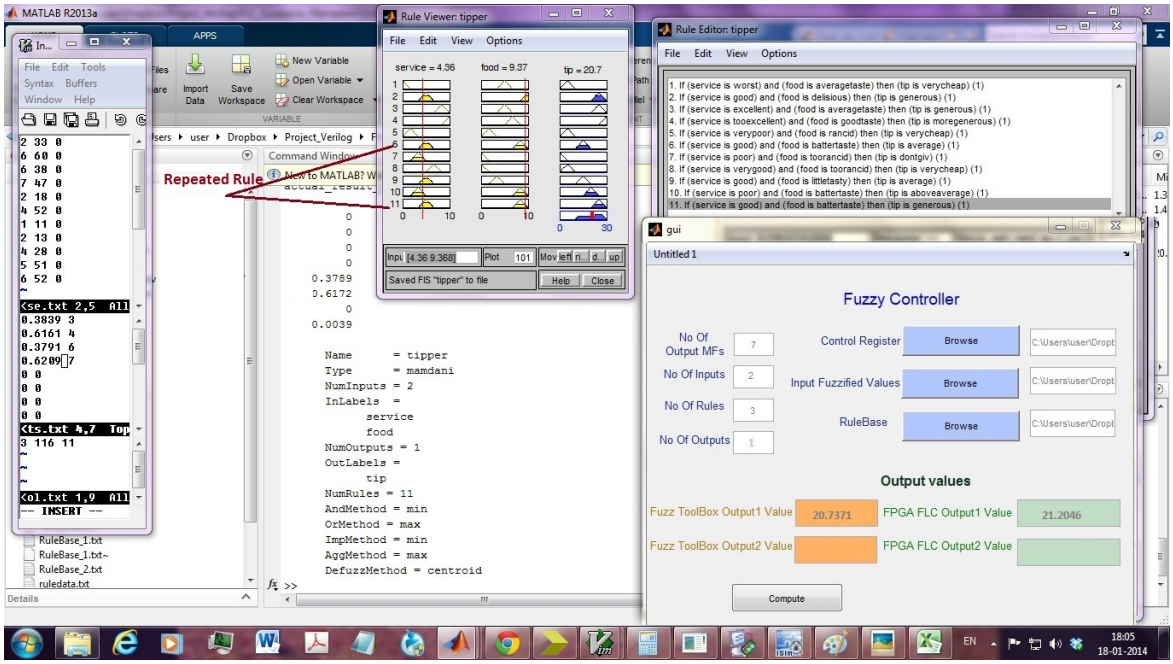


Figure 3.16: Test Model for repeated rule

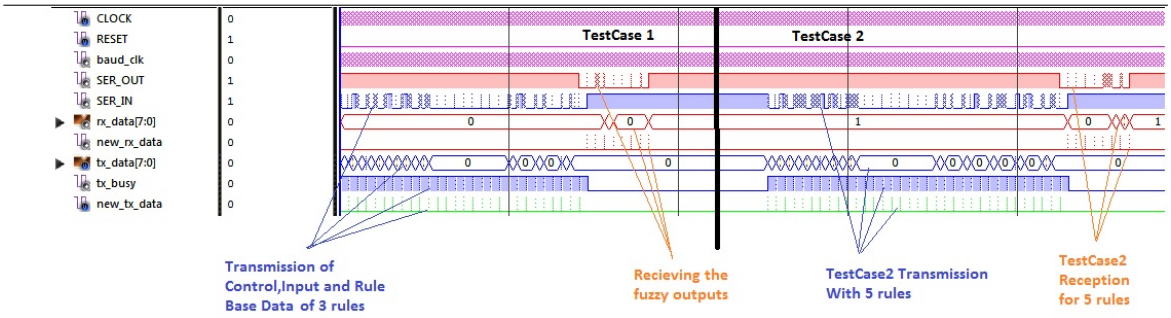


Figure 3.17: Continuous Data Transfer from DFLC to MATLAB

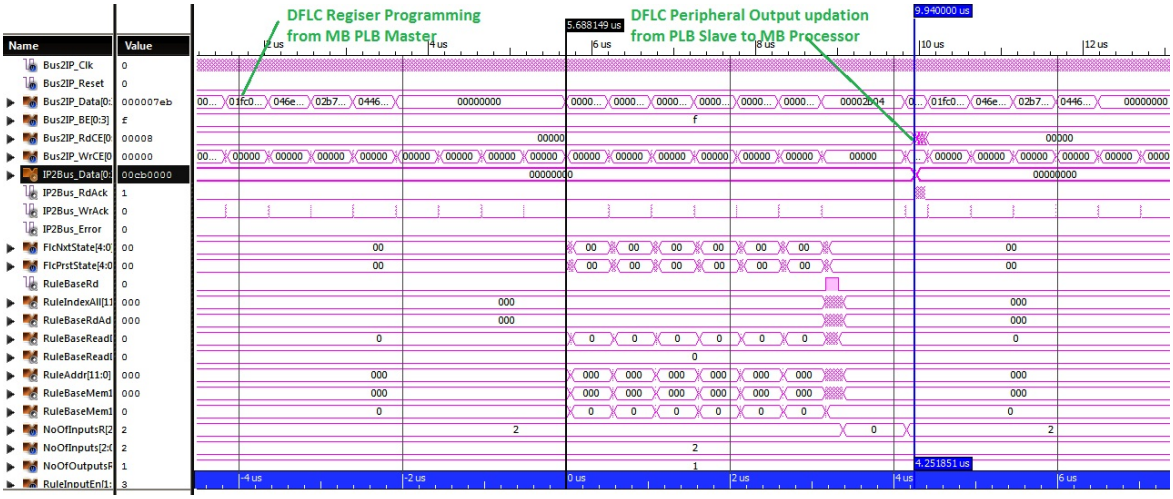


Figure 3.18: DFLC register updation from MB through PLB interface

3.5.1 Test Plan

The verification plan or test plan is a specification for the verification effort. It is used to define a first-time success, how a design is verified. Table 3.5 shows such test plan in this document to check and debug the DFLC architecture on FPGA. A fundamental 4 input 2 output tipping problem [122] with different rules and membership functions are used here to generate different test cases.

Table 3.5: Test Plan to verify the functionality of DFLC on FPGA

Test Case Name	Service	Food	Environment	Facilities	Fuzzy toolbox (Tip)	Fuzzy toolbox (Rating)	DFLC-FPGA (Tip)	DFLC-FPGA (Rating)
TDFlc2In1Out7Mf5Rul	9.5	5.95	-	-	26.348(0:30)	-	26.537(0:30)	-
TDFlc2In1Out7Mf6Rul	8.85	6.05	-	-	25.4(0:30)	-	25.7043(0:30)	-
TDFlc2In1Out7Mf9Rul	4.3	1.735	-	-	15(0:30)	-	15(0:30)	-
TDFlc2In1Out7Mf3ParRul	4.135	1.055	-	-	9.99728(0:30)	-	10(0:30)	-
TDFlc2In1Out7Mf3SFSRul	4.135	1.055	-	-	9.9972(0:30)	-	15(0:30)	-
TDFlc3In1Out7Mf3ParRul	8.012	5.482	0.584	-	20.0028(0:30)	-	20(0:30)	-
TDFlc3In1Out7Mf4ParRul	4.036	3.072	0.620	-	15 (0:30)	-	15(0:30)	-
TDFlc3In2Out7Mf3ParRul	0	5.923	0.5682	-	-9.998(-30:0)	20.0023(0:30)	-10(-30:0)	20(0:30)
TDFlc3In2Out7Mf3ParRul	1.176	0.7	0.225	-	-20.0025(-30:0)	5.00234(0:30)	-20(-30:0)	5.001(0:30)
TDFlc4In2Out7Mf7ParRul	-3.636	1.389	0.1182	0.1	-25.1062(-30:0)	22.222(0:30)	-25.150(-30:0)	22.5441(0:30)

#TFLC-Tunable Digital Fuzzy Logic Controller,xIn- x inputs, yOut-y Outputs,lMf- l Membership Functions, rRul- r Rules

3.6 System Implementation and Validation

To provide proof of concept for the proposed design, an experiment was carried out with Xilinx Virtex5 LX110T Board. The code developed for hardware DFLCs was synthesized using Xilinx ISE 14.5 to generate the DFLCs bit files for both 8bit and 16bit data widths. Two Tank Water Level Controller Plant model [123], Ball and Beam System [124] were developed in MATLAB and communication with FPGA was established using UART serial commands in a Hardware-in-the-Loop testing method. Figure 3.19 presents the central concept of the HIL test methodology, where a DFLC is developed on an FPGA and its interaction with the process plant simulation in the MATLAB. The functionality is executed in following steps:

Step1: Generation of control register set from FIS model.

Step2: Apply rule base extracted from the FIS model to GUI.

Step3: Convert all parameters into fixed point mode.

Step4: Transmit parameters from UART to FPGA by using fwrite serial command.

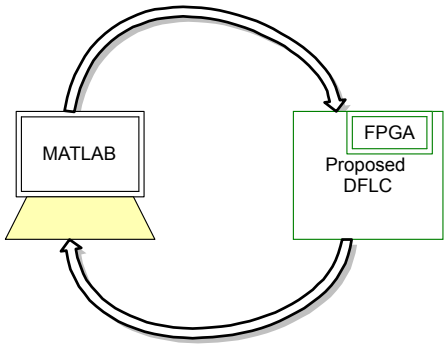
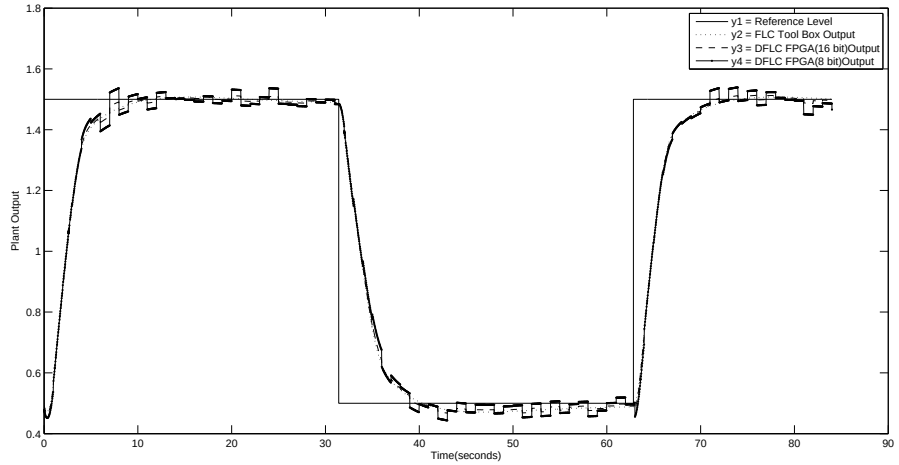


Figure 3.19: The setup for Hardware-in-loop Testing for DFLC

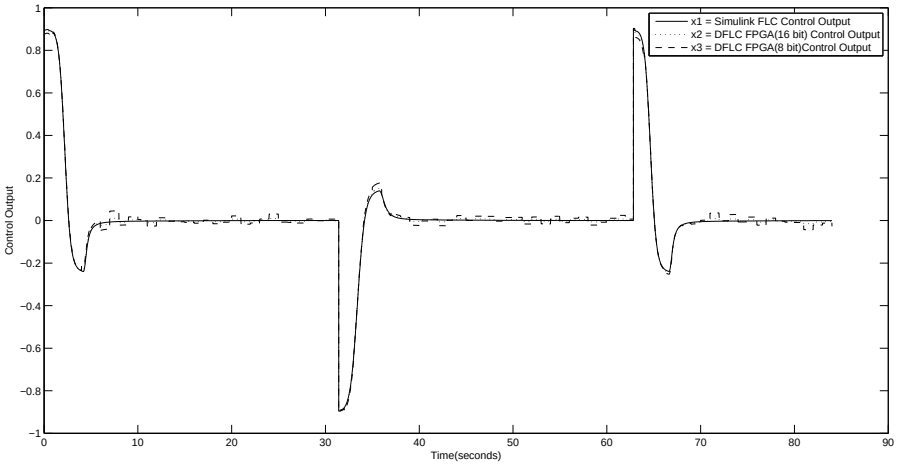
Step5: The plant model is executed with the received data from FPGA and returns the output to MATLAB. This forms the controller output.

Step6: The simulink plant output is computed with controller output. It is stored and plotted with respect to plant output using DFLC in 8 and 16 bit modes for comparative analysis.

Some benchmark control problems namely, Two Tank Water Level Controller [123], Ball and Beam System [124] were used to test the applicability and performance of the proposed DFLC architecture. In Figure 3.20 and Figure 3.21, the observed results from these simulated tests are displayed. In Figure 3.20a, the plant output of this proposed 8bit DFLC and 16 bit DFLC are compared to a MATLAB FLC controller when applied to control a Two Tank Water Level Controller [123]. The Figure 3.20 plots the plant response under these controllers with respect to time in seconds. It can be observed that the DFLC attains the performance of FIS model using the fuzzy toolbox with word length truncation error. Similar results can be obtained for Ball and Beam System [124] as shown in Figure 3.21. Figure 3.20b and Figure 3.21c compares the control output from 8bit DFLC, 16 bit DFLC and MATLAB FLT for every sample while controlling Two Tank Water Level Controller [123] and Ball and Beam System [124]. The results convincingly reflect that the proposed system architecture with its hardware implementation performs satisfactorily and can be applied on real-time. These tests also indicate that the parameters of DFLC can be remotely configurable.



(a) Plant Output: Two Tank Water Level System

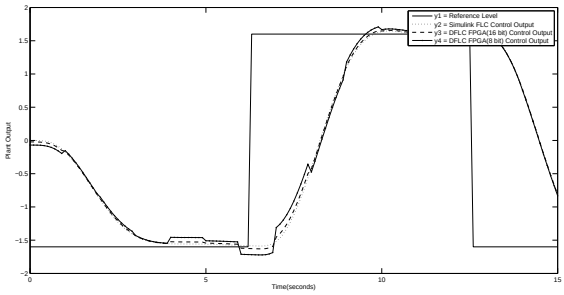


(b) Control Output: Two Tank Water Level System

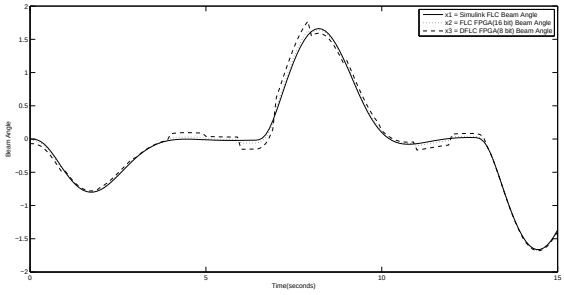
Figure 3.20: Plant and Control output of 8 bit, 16 bit and MATLAB FLT test modes using HIL for two tank water level system

3.7 Plasma Position Control in Nuclear Fusion Reactor

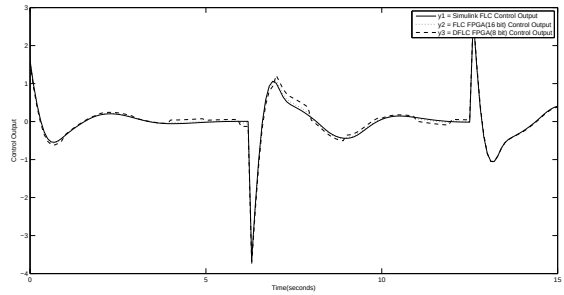
This section describes a control problem where the radial position of a plasma column in Aditya Tokamak Fusion Test Reactor (TFTR) [125] is managed. Aditya TFTR is installed at Institute of Plasma Research (IPR), Gandhinagar, India. It is a medium size tokamak with a major radius of 0.75 m and a minor radius of 0.25 m. There are 20 toroidal field coils in the design which produces maximum field strength of 1.2 tesla. A tokamak is a magnetic field based plasma confining device in the shape of a torus. Stable plasma equilibrium can be achieved by generating magnetic field lines, which can helically move around the torus. The plasma position control in a Tokamak reactor is a highly nonlinear control problem.



(a) Plant Output (Ball Position): Ball and Beam System



(b) Plant Output (Beam Angle): Ball and Beam System



(c) Control Output: Ball and Beam System

Figure 3.21: Plant and Control output of 8 bit, 16 bit and MATLAB FLT test modes using HIL for ball and beam system

In Tokamak reactor, a magnetic field is used to confine the plasma in the desired position. Plasma is highly sensitive state of matter and can be unstable under slightest trigger in the surrounding environment. It is, therefore, crucial to design a fast but highly robust controller. A tokamak can successfully operate, if the plasma is stable and confined to the geometric center of the vacuum vessel. The radial position of the confined plasma inside the torus vessel inflicts on the quality of the plasma discharge. Unstable plasma, when approaches too close to the wall of the vessel, may lead to partial or complete disruption of the plasma. Hence, it is of primal importance that the plasma position is controlled throughout the plasma discharge process.

To achieve stable plasma equilibrium and to confine it inside, a fusion reactor is required to generate a magnetic field lines that helically embraces the torus shaped plasma. These magnetic field lines can be created using electromagnets positioned suitably. Generation of helical field can be achieved by adding a magnetic field that circularly travels around the torus (toroidal field) and another field that travels orthogonally across to the toroidal field (poloidal field). These fields are generated by toroidal field coils and inner and outer poloidal field coils as shown in Figure 3.22. When a current is passed to a centrally located helical inner poloidal magnetic field coil, it produces an induced current in the plasma. The direction of the coil current and induced plasma current is shown using arrows. This plasma current generates a poloidal magnetic field. The required toroidal magnetic field is produced by the circularly surrounded coils across the torus. The position of the plasma can be controlled by driving the electric current to these coils.

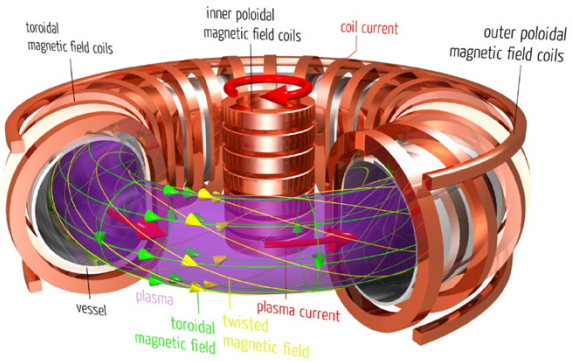


Figure 3.22: Schematic of a tokamak.
Photo credit: Abteilung Öffentlichkeitsarbeit - Max-Planck Institut für Plasmaphysik. Licensed under Creative Commons BY–SA 3.0 via Wikimedia Commons

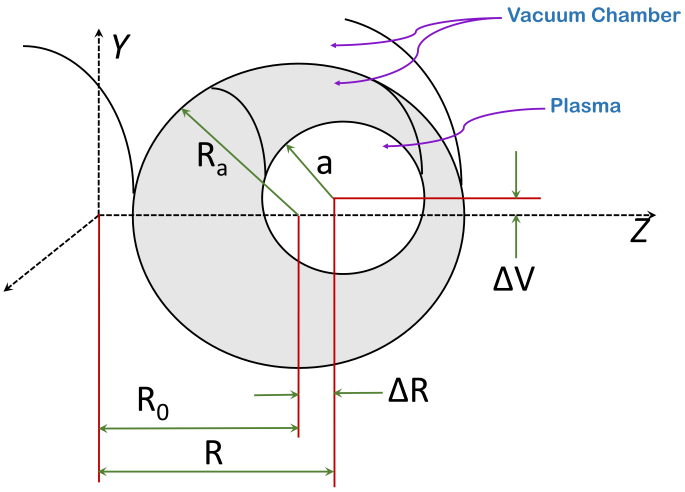


Figure 3.23: Cross-sectional view of plasma position and displacement inside the vacuum chamber

Table 3.6: List of Variables

I_p	Plasma current	μ_o	Permeability of Vaccum
V_c	Control voltage	β_p	Poloidal beta
I_c	Coil and conductor current	Γ	Shafranov parameter
RI_p	Weighted plasma radial position	B_v	Vertical magnetic field
zI_p	Weighted plasma vertical position	l_i	Internal inductance of plasma magnetic field
R	Major Radius	a	Minor radius

3.7.1 Aditya Tokamak System Modeling

In this section, the control of the radial position of plasma in Aditya TFTR using fast feedback (FF) coil has been analyzed using an RZIP model. The geometric center of the vacuum vessel of Aditya TFTR is at 0.75 m, and it is critical that the radial position of the plasma is maintained at this point. This model is developed with the assumption that small variation in coil currents produces a small change in plasma position and current. Table 3.6 lists the variables in the model.

Unlike circular cross-section plasma, Tokamak operates on highly non-circular torus shape. Non-circular shapes are harder to generate and to control accurately since currents through several control coils must be adjusted simultaneously [126]. Due to uncertainties in the current and pressure distributions within the plasma, the desired accuracy for plasma control can only be achieved by making real-time measurements of the position and shape of the boundary, and using error feedback to adjust the currents in the control coils. The modeling of the discharge parameters like plasma current, position and shape is a challenging task, as they are highly nonlinear and time varying in nature. Hence, it is hard to achieve control of plasma position using traditional controllers [1] due to the inherent complexity of the plasma position control system and its nonlinear nature. A similar approach was also taken by Morelli *et. al.* [127] in plasma position control of STOR-M Tokamak Fusion Test Reactor.

Considering the above modeling parameters taken from the work by Bandyopadhyay *et. al.* [128, 129],

$$\dot{X} = \mathbf{A}X + \mathbf{B}U \tag{3.21}$$

Where, $\mathbf{A} = M^{-1} \cdot R$, $\mathbf{B} = M^{-1}$, and

$$X = \begin{bmatrix} I_C \\ zI_P \\ RI_P \\ I_P \end{bmatrix}, U = \begin{bmatrix} V_C \\ 0 \\ -\frac{\mu_0 I_P}{2}\Gamma \\ I_P \end{bmatrix}$$

M and R refers to vector of mutual inductances and resistances of all circuits with plasma [1, 129, 130].

$$M = \begin{bmatrix} M_C & (M'_R)^T & M_{PC} \\ M'_Z & 0 & 0 \\ M'_R & M_{22} & M_{22} \\ M_{PC} & M_{22} & L_{P0} \end{bmatrix} \text{ and } R = \begin{bmatrix} \Omega_C & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Omega'_P & \Omega_P \end{bmatrix}$$

where,

$M_{22} = \left(\frac{\mu_0}{2}\frac{d\Gamma}{dr} + \frac{2\pi B_{z0}}{I_{p0}} + \frac{2\pi R_0 B'_{z0}}{I_{p0}}\right)$

$M_{23} = \left(\mu_0\Gamma_0 + \frac{2\pi R_0 B_{z0}}{I_{p0}}\right)$

$M_{32} = \left(\mu_0\left(1 + f_0\right) + \frac{2\pi R_0 B_{z0}}{I_{p0}}\right)$

Where, M_C and Ω_C are mutual inductance and resistance matrices of all the circuits, M_{PC} and M_R are the vector of mutual inductances of the circuits with the plasma and their radial derivatives respectively, and Γ is known as Shafranov parameter. This shows that A and B are matrices that are dependent on the mutual inductances and resistances of all circuits with plasma, which is highly non-linear in nature.

In Aditya TFTR, four magnetic probes are used to measure the radial position of the plasma. These probes are placed close to the outer periphery of the vacuum vessel. A Rogowski coil is used to measure the plasma current (I_P).

P. Suratia *et. al.* [1] and I. Bandyopadhyay *et. al.* [130] explained the major control operatives as - 'ADITYA has been provided with a primary vertical coil field with an adjustable gain proportional to plasma current, to compensate the change in vertical displacement of plasma column. The shift in radial position due to minor disruptions is controlled by a separate pair of Fast Feedback coils, and this fast feedback coil produces an adequate magnetic field to bring the plasma column back to its geometrical center'.

3.8 Control Strategy

3.8.1 Using PID Control

Traditional PID controllers [131] are used presently in Aditya TFTR to which radial position signal is fed as input. The controller generates a suitable control signal to actuate the current in the fast feedback coils and accordingly the plasma is confined in radial direction [1]. Figure 3.24 shows the control strategy employed in radial position control of plasma in Aditya TFTR.

$$\frac{\Delta B_v}{B_v} = \frac{\Delta R}{R}$$

The vertical field required to maintain the radial position of plasma in Aditya TFTR can be obtained from Grad-Shafranov equation [1, 132] presented at (3.22).

$$B_v = \frac{\mu_0 I_P}{4\pi R}\left[\ln\left(\frac{8R}{a}\right) + \beta_P + \frac{l_i - 3}{2}\right] \tag{3.22}$$

where, B_v is Magnetic flux density, B_ϕ, B_θ, B_ρ is Toroidal, poloidal and radial components of the magnetic field, E is Electric field intensity, J is Plasma current density, R is Major radial coordinate and a is Minor radial coordinate. It can be observed from (3.22) that, the total vertical magnetic field for proper position control of plasma is proportional to the magnitude of

1. Internal inductance of plasma l_i,
2. Plasma current I_P, and
3. Plasma poloidal beta { It is the ratio of the poloidal plasma pressure to the poloidal magnetic pressure}.

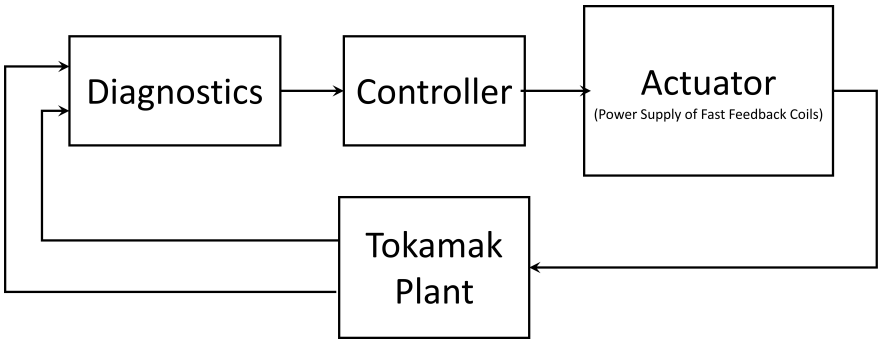


Figure 3.24: Control strategy for radial plasma position control in Aditya TFTR

All these parameters are time varying and highly nonlinear in nature. The limitations of PID controllers have been already explained in section 1.5, and therefore, a controller equipped to handle these parameters to provide a smooth, fast and robust control action is of utmost importance. However, it is important to exercise the existing knowledge gathered from the system response with PID controller. A PID control loop for radial plasma position control of Aditya TFTR is developed in Simulink as shown in Figure 3.25. It represents the mathematical model explained in (3.22). The PID controller is tuned using Ziegler-Nichols method. The Ziegler-Nichols tuning method is a heuristic method of tuning a PID controller [129, 130]. The simulation output data is observed and recorded.

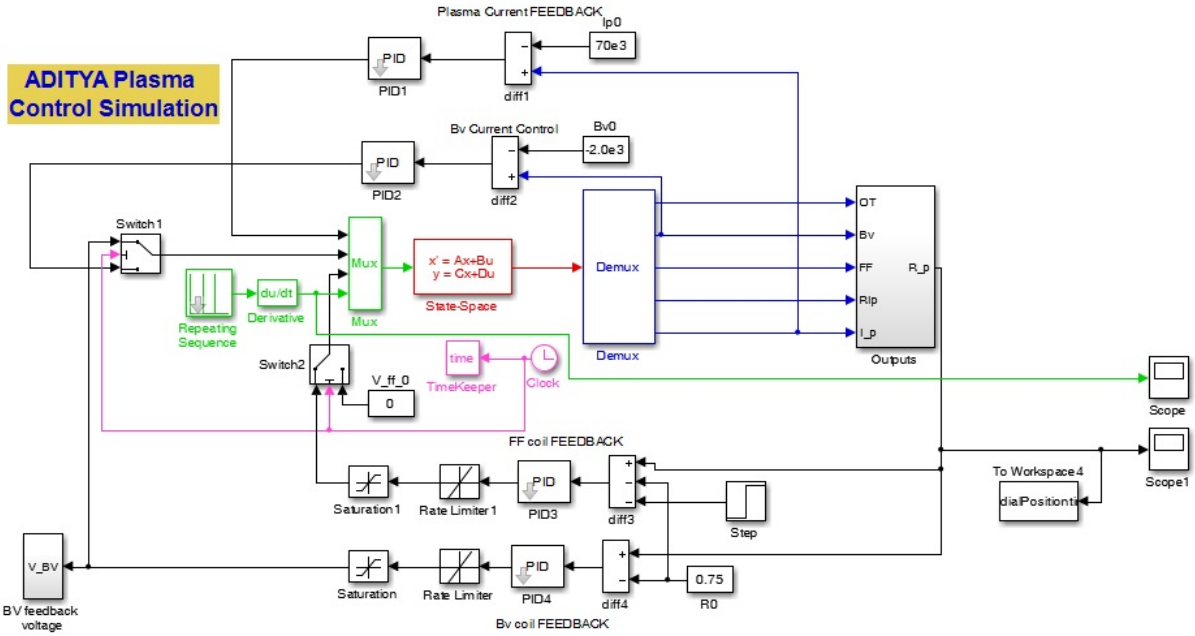


Figure 3.25: Simulink model of radial plasma position control in Aditya TFTR with PID controller

3.8.2 Plasma Position Control in Aditya using FLC and DFLC

P. Suratia *et. al.* proposed a fuzzy logic controller for radial plasma position control in Aditya TFTR [1]. The characteristic features of this controller and the proposed DFLC used in the control simulation is tabulated in Table 3.7. The simulink model for the control of the system with FLC is shown in Figure 3.26. It can be observed in this figure that the inner PID loop in the outer position control loop in Figure 3.25 is replaced by an FLC loop in Figure 3.26.

The DFLC is connected to a PC with Simulink model of radial plasma position control of Aditya TFTR. Using UART, data can be exchanged between the two systems. The hardware DFLC polls for any input at the UART. Once it receives the input, it completes the FLC with

Table 3.7: Characteristics of FLCs used in [1] and DFLCS

Parameters	[1]	DFLC
Inputs	2	2
Output	1	1
Antecedent MFs	7 (triangular)	7 (triangular)
Consequent MFs	7 (singleton)	7 (triangular)
Aggregation	MIN	MIN
Implication	MAX	MAX
MF Overlapping Degree	2	Dynamic (4)
Defuzzification Method	Weighted Average	Weighted Average

FLC execution process to return a suitable control signal to the power supply of the feedback output. This completes hardware in loop and it is continued for the entire simulation. Simulation for other controllers is carried out sequentially by changing the manual switches as shown in Figure 3.26. These simulation data are recorded for all control schemes and analyzed for the performance of the controllers.

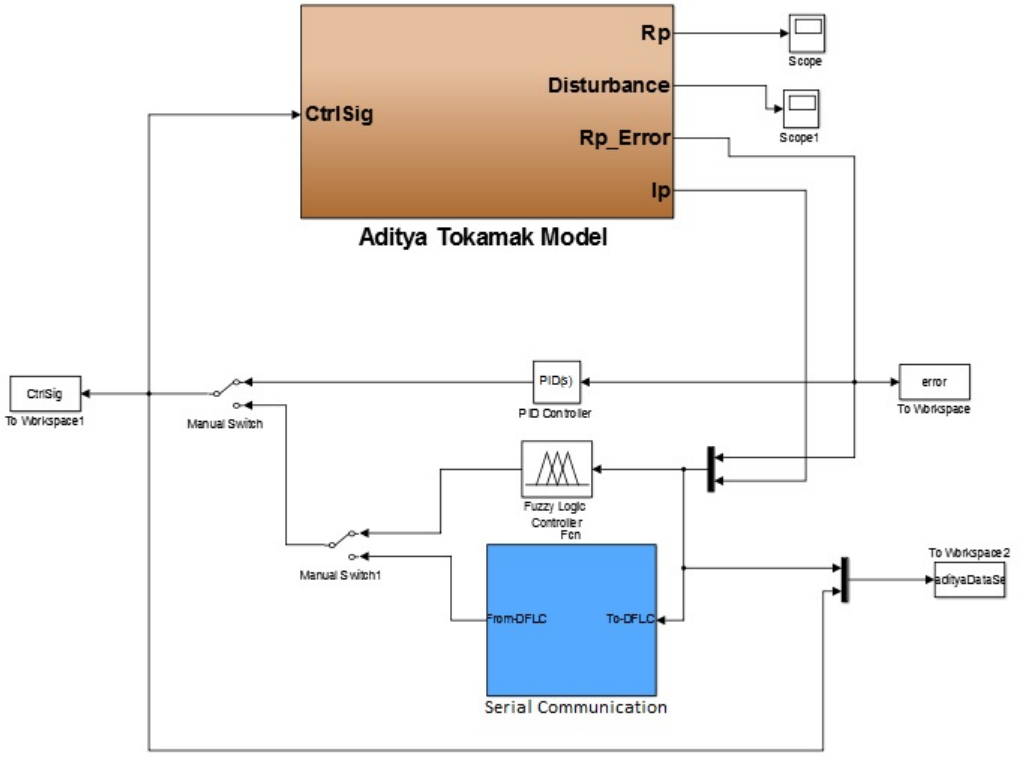


Figure 3.26: Simulink model of radial plasma position control in Aditya TFTR with FLC and DFLC

The recorded data from the simulations is plotted as depicted in Figure 3.27. This plot clearly displays the difference in the control action. A significant improvement in rise time and settling time is observed in accordance to the PID controller and existing

FLC. The hardware DFLC is observed to provide a smooth and fast response. It caters a robust control scheme for the radial position control in Aditya TFTR. A comparative analysis of the control parameters is drawn and tabulated in Table3.8. It can be observed that the DFLC with MRA2-OMF method provided 62% faster rise time and 80% speedy settling time, MRA3-OMF method provided 62% faster rise time and 82% speedy settling time, and MRA4-OMF method provided 64% faster rise time and 83% speedy settling time in comparison to existing control schemes. The computational complexity comparison is provided in Table 3.9, Which shows the logic utilization is increasing from MRA-2-OMF to MRA-3-OMF and MRA-3-OMF to MRA-4-OMF.

Table 3.8: Comparison of performance parameters of PID, FLC [1], and DFLC with MRA2-OMF, MRA3-OMF and MRA4-OMF Methods

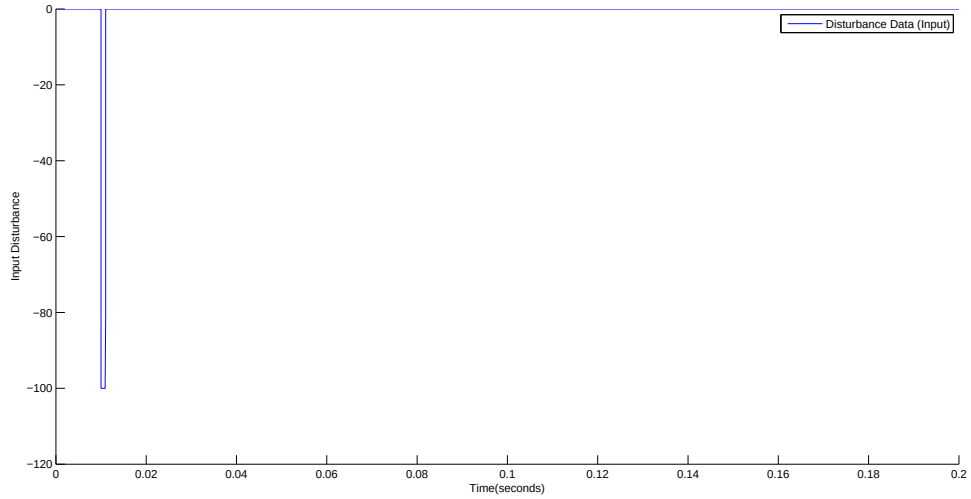
Parameters	PID	FLC [1]	DFLC (MRA2-OMF)	DFLC (MRA3-OMF)	DFLC (MRA4-OMF)
Rise Time	0.0062	0.0062	0.0025	0.0023	0.0022
Settling Time	0.1255	NaN	0.0249	0.0223	0.0213
Overshoot	0	0	0	0	0
Undershoot	0	0	0	0	0
Peak	0.7497	0.7483	0.7502	0.7487	0.7487
Peak Time	0.14	0.02	0.0235	0.0230	0.0228

3.9 Summary

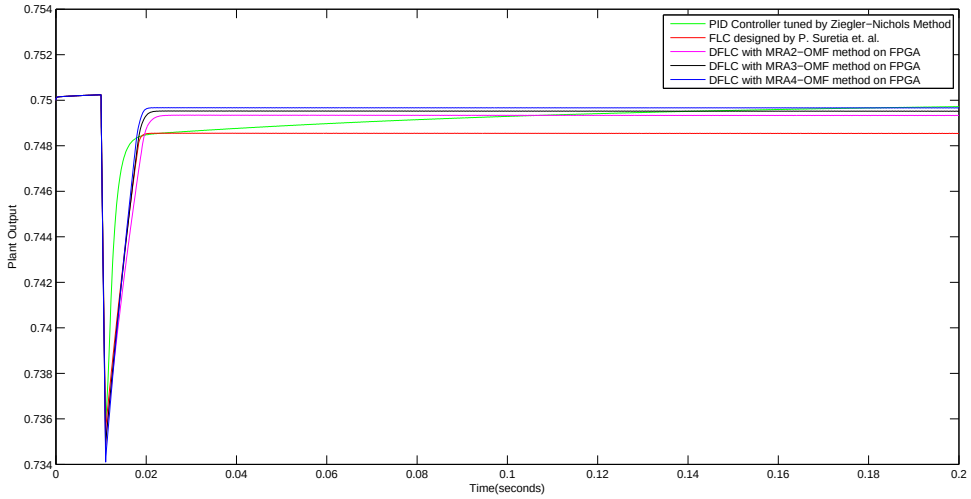
This chapter presented a novel system architecture for a general purpose fuzzy logic controller based on FPGA. The proposed FPGA-based FLC can be used for any control application effectively. Here, the idea is to implement a flexible algorithm running on hardware which can be configured remotely on user demand. This chapter elaborated proposed MRA-3OMF and MRA4-OMF rule reduction technique with a special case rule base state machine support. The tunability of FLC parameters is provided in the DFLC architecture as mentioned in section 3.4. GUI based fuzzy validation with its test vectors is analyzed. MB processor based IP integration with its simulation shows that the proposed DFLC can be easily connected as peripheral to any soft or hard processor with processor

Table 3.9: Computational Complexity of all proposed methods

Proposed Methods	BRAM utilized		FFs	LUTs	Cycle Time	Latency
	32K	18K				
MRA2-OMF	0	1	992	2327	6.608 ns	99.12 ns
MRA3-OMF	0	1	1154	2551	6.754 ns	101.31 ns
MRA4-OMF	0	1	1386	2613	6.913 ns	103.695 ns



(a) Input Disturbance Signal



(b) System Response

Figure 3.27: Performance of various controllers in presence of disturbances in plasma position

logic bus. The proposed DFLC can suitably replace existing controllers in a process plant that confirms the generic nature of the designed DFLCs. The applicability of the proposed method was also tested by applying it to a benchmark problem. The results portrayed a proof-of-concept for the objectives that were set in chapter 1. The observation obtained from this system was exciting as it provided around 60% faster rise time and around 80% speedy settling time in comparison to existing control schemes. These results are extremely positive and encouraging.

Chapter 4

Tunable Type 2 Fuzzy Logic Controller with Successive Approximation based Membership Function

Preface

In this Chapter, the Type 2 fuzzy system with dynamic digital type 2 fuzzifier, using a successive approximation is presented. The hardware implementation of the proposed Type 2 FLC is characterized by online tunability, choice of rule reduction technique, and partial rule base support. The Successive Approximation based Iterative Type 2 Fuzzy Logic Controller (SAIT2FLC) is observed to have a cycle time of 267.9 ns i.e., 3.7 MFLIPS for 2 input 4 rule IT2FLC system. A UART interface is proposed for the communication between MATLAB IT2FLC GUI and Virtex LX110TFPGA for system level testing. In this chapter, the proposed SAIT2FLC is used to control the radial plasma position in Aditya TFTR.

4.1 Introduction

When something is uncertain, as a measurement, it's hard to determine its exact value and of course, type 1 fuzzy sets make more sense than using crisp sets. However, it is not reasonable to use an accurate membership function, if the fuzzy rules are uncertain. Such uncertainty leads to rules, whose antecedents or consequents are uncertain, which translates into uncertain antecedent or consequent membership functions [133]. In this case, we need another type of fuzzy sets to handle these uncertainties called type 2 fuzzy sets. Iterative Type 2 Fuzzy Logic Controllers (IT2FLC) [134, 135] with its membership functions in type 2 fuzzy sets [136, 137] can handle rule uncertainties (characterized by more parameters) than their predecessors Type-1 Fuzzy Logic Controllers (T1FLC) [138]. In IT2FLC uncertainties are accommodated by using Upper Membership Function (UMF) and Lower Membership Function (LMF) of two type 1 fuzzy sets [4], called Footprint of Uncertainty (FOU). There have been many applications that have shown that IT2FLCs perform well compared to its predecessor T1FLC and some traditional controllers.

Type 2 Fuzzy Logic Controller (T2FLC) implementations in hardware have been of current research interest [139–142] with different control applications [143, 144]. There are analog, digital, and microprocessor based T2FLCs implementation in literature [145–147]. In these implementations, there has been limited discussion on the design aspects of type 2 fuzzifier, especially, digital type 2 fuzzifier. Usually, fuzzification is implemented either by software or by Look-Up-Table (LUT) method. The first method, in general, is not suitable for real-time applications, and the latter is not appropriate considering its computational complexity. The other competing method for hardware realization is to compute membership function by using arithmetic circuits. This approach is characterized by complexity that is being related to the type of membership functions. Triangular and Trapezoidal membership functions are computationally secured than other membership functions because of their first order polynomial implementation. Hence, these two types of membership functions have been used to realize in hardware.

Analog implementations have been implemented by different designs methodologies, Where A. Mesri *et. al.* [148] presented a fully programmable IT2 membership function generator using slope tuning method. By getting required current biasing from the voltage to current converter, the membership function was developed by K.P. Abdulla *et. al.* [149]. However, the analog nature of these circuits limit the accuracy of the fuzzifier, and it also

needs extra circuits like A/D and D/A converters. In these circumstances, an improved digital fuzzifier serves better with its advances in digital implementations using reconfigurable hardware.

In this chapter, a hardware implementation of type 2 fuzzifier using successive approximation is proposed; the parameters can be tuned and configured by the upper layer. This fuzzifier supports trapezoid and triangular membership functions with nine parameters represented by P1, P2, P3, P4, P5, P6, P7, P8, and P9 points as shown in Figure 4.3b. We choose 16-bit Word Length (WL) in this design to acquire good accuracy. Since the inference mechanism in T2FLC and T1FLC are similar, it is applied here the same rule reduction methods and special case rule support of rule base as in section 3.4.1. The significant hardware complexity involves in T2FLC is in its type reduction stage than the T1FLC. The popular Karnik-Mendel (KM) [150] iterative method is used extensively by researchers in the fuzzy domain. Due to its iterative nature, its hardware realization is complex with high computational cost. Its design consumes lot of resources in FPGA implementation. The primary objective of defuzzification is to find a unique output value following the inference graph. So, it only demands that there must be a relation between the shape of the inference graph and the defuzzified value. If there is a method other than the classic centroid methods, providing satisfactory results than KM method can be used as type reducer. One such method is Wu-Mendel (WM) [151] closed form method. In this work, the proposed Wu-Mendel (WM) [151] type reduction method has been adopted and implemented in hardware. The proposed reduced active rules applied on this method to obtain the crisp output.

4.2 Type 2 Fuzzy Logic Systems - An Overview

4.2.1 Type 2 Fuzzy Sets

An example of a type-2 fuzzy set $\tilde{A}$ is shown in Figure 4.1, where the membership of type-2 fuzzy set is an interval. The concept of a type-2 fuzzy set was introduced by Zadeh [152] as an extension of the concept of a type-1 fuzzy set. A type-2 fuzzy set is characterized by a fuzzy membership function, i.e. the membership grade for each element of this set is a fuzzy set in [0, 1], unlike a type-1 set, where a membership grade is a crisp number in [0, 1], type-2 fuzzy sets can be used in situations, where the uncertainty in the shape of membership function or some of its parameters. In real world problems, it can be considered type 1 fuzzy set as first order approximation and type-2 fuzzy set as second order approximation.

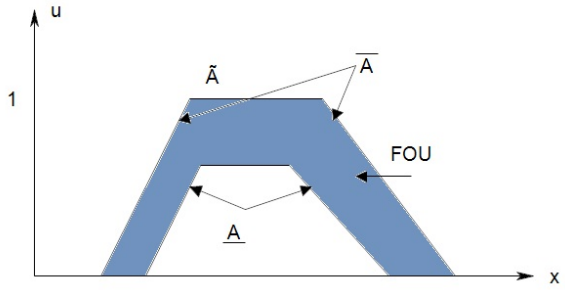


Figure 4.1: Type 2 Fuzzy Set

A type-2 fuzzy set denoted as $\tilde{A}$, is characterized by a type-2 membership function:

$$\tilde{A} = ((x,u), \mu_{\tilde{A}}(x,u))|\forall x \epsilon X, u \epsilon J_x \subseteq [0,1] \tag{4.1}$$

Here $0 \leqslant \mu_{\tilde{A}}(x,u)) \leqslant 1$, in fact $J_x \subseteq [0,1]$ represents the primary membership of x and $\mu_{\tilde{A}}(x,u))$ is a type 1 fuzzy set known as a secondary set [153]. Hence, a type-2 membership grade can be any subset in [0,1], the primary membership. Corresponding to each primary membership, There is a secondary membership (which can also be in [0, 1]). Uncertainty is represented by a region, which is called the footprint of uncertainty (FOU). An upper membership function and lower membership functions are two type-1 membership functions that bound for the FOU of a type-2 fuzzy set Ã. The upper membership function is associated with the upper bound of $FOU(\tilde{A})$ or $\bar{A}$. The lower membership function is associated with, the lower bound of $FOU(\tilde{A})$ or $\underline{A}$.

4.2.2 Type 2 Fuzzy Set Operations

To compute the union, intersection, and complement of type-2 fuzzy sets, it needs to extend the binary operations of minimum (or product) and maximum, and the unary operation of negation, from crisp numbers to type-1 fuzzy sets, because at each x, $\mu\tilde{A}_i$(x, u) is a function. Consider two type-2 fuzzy sets $\tilde{A}_1$ and $\tilde{A}_2$, i.e.,

$$\tilde{A}_1 = \int_x \mu_{\tilde{A}_1}(x)/x \tag{4.2}$$

$$\tilde{A}_2 = \int_x \mu_{\tilde{A}_2}(x)/x \tag{4.3}$$

The union of $\tilde{A}_1$ and $\tilde{A}_2$ is another type-2 fuzzy set and is named as "join" operation [154]. More formally, has the following expression.

$$\tilde{A}_1 \cup \tilde{A}_2 = \int_{x \epsilon X} \mu_{\tilde{A}_1 \cup \tilde{A}_2}(x)/x \tag{4.4}$$

The intersection of $\tilde{A}_1$ and $\tilde{A}_2$ is another type-2 fuzzy set and is named as "meet" operation [154]. More formally, has the following expression:

$$\tilde{A}_1 \cap \tilde{A}_2 = \int_{x \epsilon X} \mu_{\tilde{A}_1 \cap \tilde{A}_2}(x)/x \tag{4.5}$$

The complement of set $\tilde{A}$ is another type-2 fuzzy set, just as the complement of type-1 fuzzy set A is another type-1 fuzzy set. More formally has,

$$\tilde{A}' = \int_x \mu_{\tilde{A}'}(x)/x \tag{4.6}$$

4.2.3 Type 2 Fuzzy Logic Controllers

The primary structure of type 2 fuzzy logic system does not change from type 1 fuzzy logic system since the core principles of fuzzy logic are independent of the nature of membership function. The rule of inference like generalized modes pones continuous to apply. The general structure of type 2 FLC is shown in Figure 4.2. Here, the significant structural difference is the defuzzifier block in type 1 FLC is replaced by output processing block, which consists type reduction followed by defuzzifier. Hence the type 2 FLC comprises the following two extra modules:

i. Type Reducer: This module converts output inference engine, which is type 2 fuzzy sets into type 1 fuzzy sets.

ii. Output Processor: It consists of Type reducer followed by defuzzifier.

4.3 Successive Approximation Type 2 Membership Function

This section analyzes the digital type 2 fuzzifier shown in Figure 4.3b is described. Where, the values of the fuzzifier output are considered in 16 bit, here '1' is represented as FFFFH since maximum fuzzifier value is FFFFH. The Type 2 fuzzifier's basic operation is shown

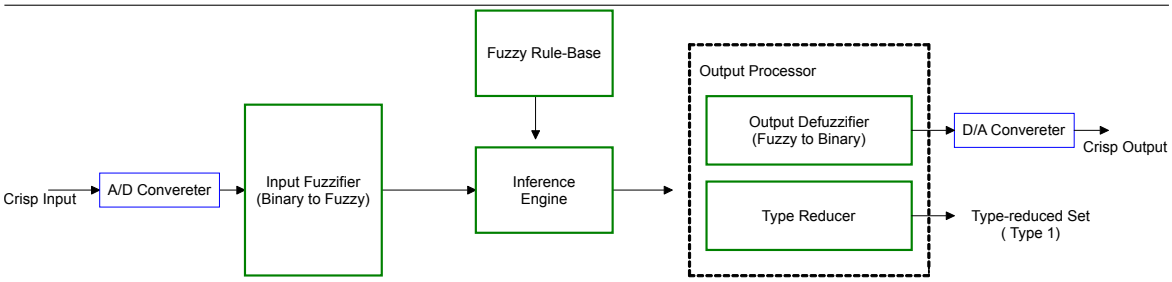


Figure 4.2: An Interval Type 2 Fuzzy Set

in Figure 4.4. It can be observed that two fuzzifications happen simultaneously from upper and lower membership functions with subtractions and divisions. It is observed that the UMF calculation is similar to type 1 fuzzier (shown in Figure 4.3a) and provides the membership value $\leq$ 1. For LMF calculation, a slight modification in the design is incorporated by multiplying height (point P9) with fuzzy calculated value. The basic operations of fuzzification are integer subtraction and division as depicted in (2.1). Since division is evolved in fuzzification, implementing high speed and the accurate divider is the major concern in type 2 fuzzification.

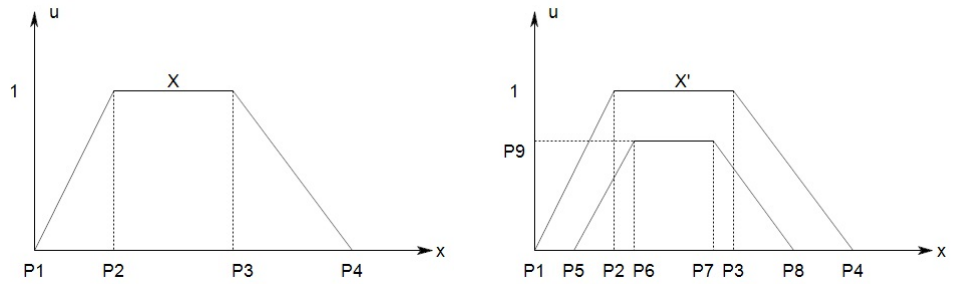
Implementation of division algorithm had an extensive literature, where digit recurrence [155–157], functional iteration [158, 159], non-restoring [160, 161], very high radix [162], newton raphson approximation [163], variable latency [164, 165] and table look-up [166, 167] are some of the implementation techniques. These are well-known implementations with their advantages with low latency, less cycle time and the support for negative integer numbers. Pipelining had been used at the cost of an area to reduce latency in the division algorithms. Using one of these methods to implement a fuzzifier is usually not preferred because of their computational complexity such as radix operations, conversion of a negative digit to binary forms and the usage of underlying multipliers. A simple successive approximation divider is proposed in this chapter to calculate membership values considering the following limitations in fuzzification.

1. The denominator (dividend) is always less than the numerator (divisor)

2. Unsigned integer division.

3. Both dividend and divisor have an equal number of bits.

Considering the above limitations as an advantage, this chapter proposed a simple division algorithm, especially for fuzzification purpose. The algorithm for a 2n radix runs



(a) 4 point representation of Type 1 Fuzzifier for one group of membership

(b) 9 point representation of Type 2 Fuzzifier for one group of membership

Figure 4.3: Trapezoidal Type 1 and Type 2 Fuzzifiers

in following steps:

Step 1: Load initial values of dividend and divisor to the divider input.

Step 2: Segregate divisor in to n regions (Equally) as depicted in Figure 4.5, Where, Figure 4.5a presents the segregation of eight equal regions in Left Shoulder Left Foot (LSLF) and Figure 4.5a presents the segregation of eight equal regions in Right Shoulder Right Foot (RSRF).

Step 3: Compare the dividend value with each region. Assign the region index values at Y-Axis of Figure 4.5a to the 3 bit MSB of quotient value.

Step 4: Subtract the regions lower bound value from divisor and dividend. Left shift 3 times and assign the new region index value to the 3 bit MSB of quotient value.

Step 5: Repeat 'Step 2' for newer values of divisor and dividend as given in Figure 4.6. Where, the input at the region $3/8^{th}$ of input value to $1/4^{th}$ of the input value is expanded for further iteration.

Step 6: Repeat the steps from 'Step 2' to 'Step 5' until a proper precision is reached.

Step 7: Assign final quotient value with appropriate enable signal for validating its usage in next module.

Figure 4.7 presents the digital logic circuit implementation of membership circuit. This system consists of the priority encoder, subtractor, edge detector, shifter and a multiplexer. In this design, the membership values use 16 bit representation, where the values of "0000H" = 0 and "FFFF" = 1 with precision of 0.0000152. 'Shift+ Add' module generates eight

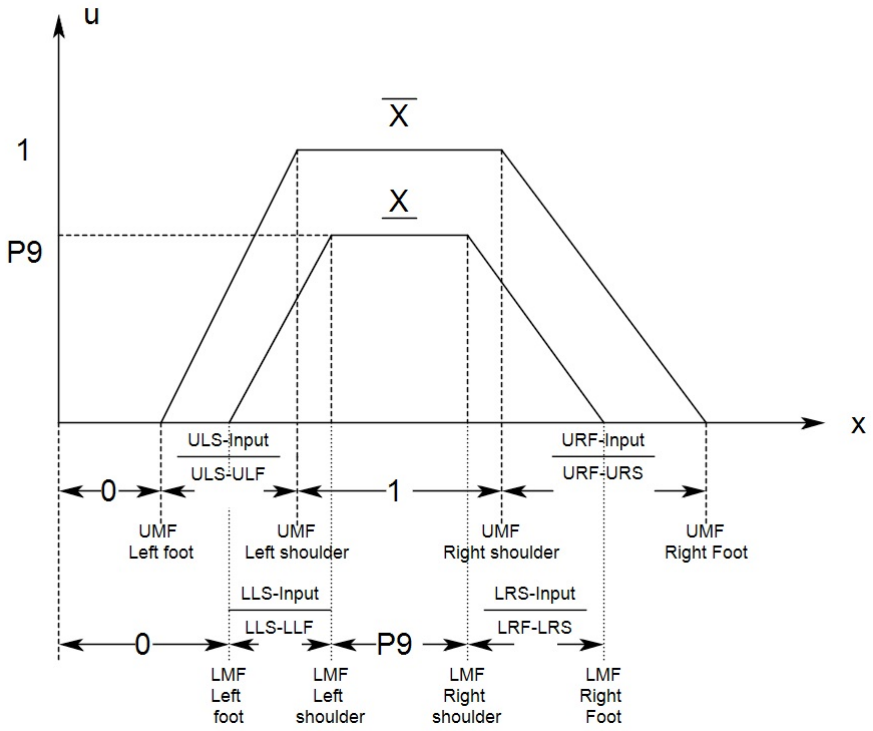


Figure 4.4: Basic operation of Type 2 Fuzzification

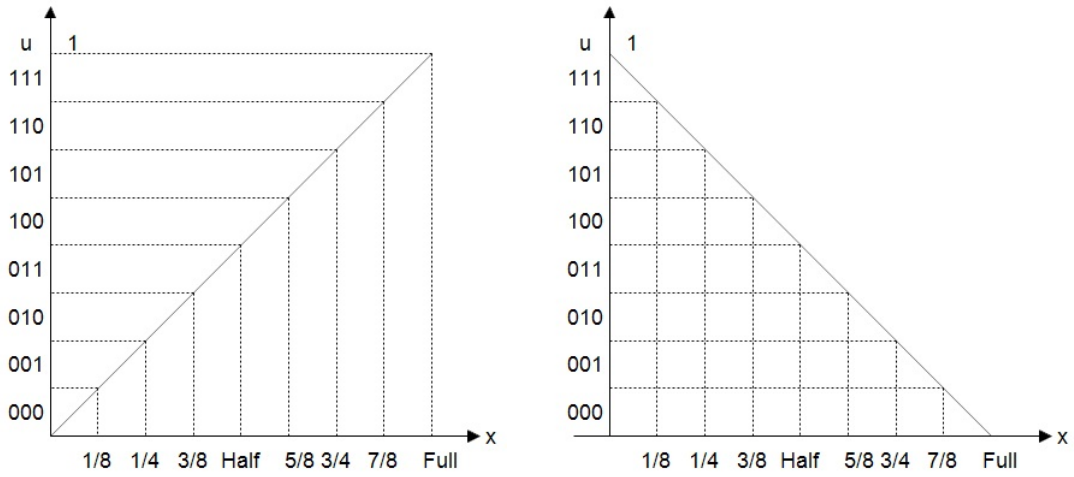
regions from divisor. 'Comparator + Subtractor' module compares the dividend value with in each region and subtracts its lower bound from divisor and dividend. 'Membership Values Generator' generates quotient by using shift operation. Multiplexer selects and places the newly made dividend and divisor values. The counter controls the precision of the output.

The circuit models of lower and upper membership functions are depicted in Figure 4.8 and Figure 4.9 respectively. In these circuits the comparator compares the input to find its membership value from following 5 cases:

Case 1: If the input value is less than UMF and LMF left foot, the circuits result in zero membership value.

Case 2: If the input value is between the UMF left foot and UMF left shoulder or LMF left foot and LMF left shoulder, The comparator selects InputX-P5, P6-P5 values as dividend, divisor values for UMF circuit and InputX-P1, P2-P1 values as dividend, divisor values for LMF circuit.

Case 3: If the input value is between the UMF left shoulder and UMF right shoulder or LMF left shoulder and LMF right shoulder, the membership value is taken as '1' in the case of UMF Circuit and P9 in the case of LMF Circuit.



(a) Left Shoulder-Left Foot (b) Right Shoulder-Right Foot

Figure 4.5: Basic function of membership circuit

Case 4: If the input value is between the UMF/LMF right shoulder and UMF/LMF right foot, the comparator selects P8-InputX, P8-P7 values as dividend, divisor values for UMF circuit and P4-InputX, P4-P3 values as dividend, divisor values for LMF circuit.

Case 5: If the input value is greater than UMF right foot or LMF right foot, the circuits result in zero membership value.

The top-level block of type 2 fuzzifier with successive approximation based upper and lower memberships is given in Figure 4.10. Where, the LMF circuit uses extra input to accommodate point P9. Internal digital structure of Interval Type 2 Successive Approximation based Higher Membership Function (IT2SAHMF) and Interval Type 2 Successive Approximation based Lower Membership Function (IT2SALMF) are presented in Figure 4.8 and Figure 4.9 respectively. Where, membership circuit is used as a divider to support Case 2 and Case 3.

4.4 Tunable Type 2 Fuzzy Logic Controller

4.4.1 Digital Architecture

This section presents the architecture of Type 2 FLC for hardware implementation. Similar kind of interface as discussed in section 3.4.2 is established using the UART module to configure the Type 2 FLC from MATLAB GUI. The top level architecture of SAIT2FLC is presented in Figure 4.11. Where, the proposed successive approximation based type 2

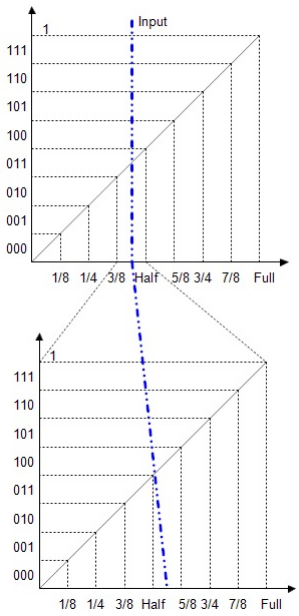


Figure 4.6: Algorithm flow of successive approximation

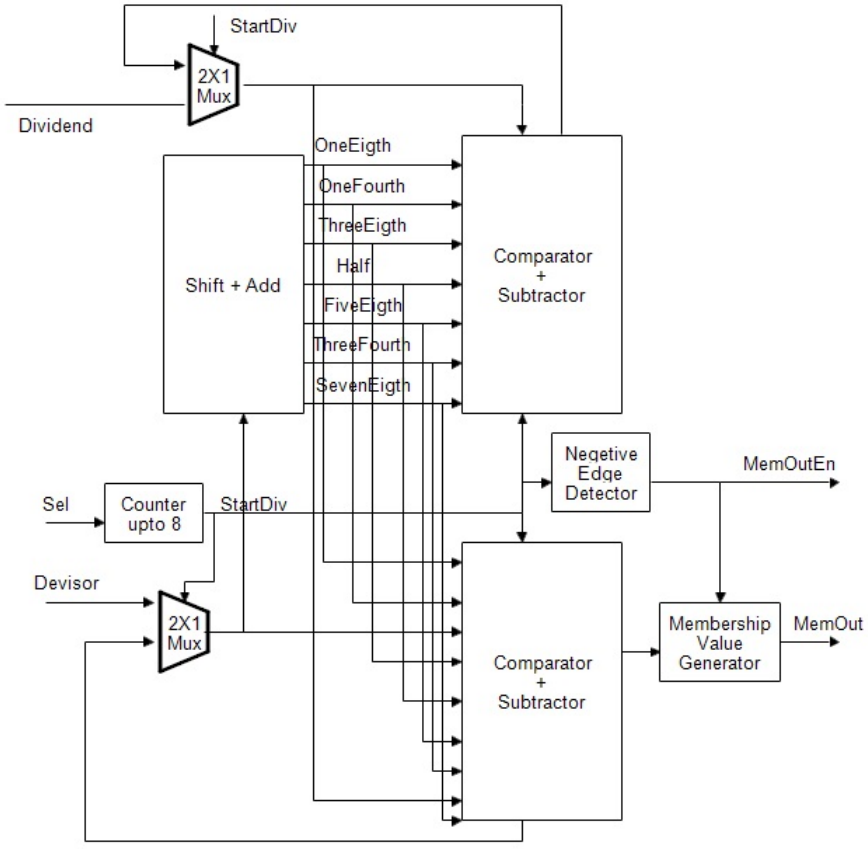


Figure 4.7: Design of membership circuit module

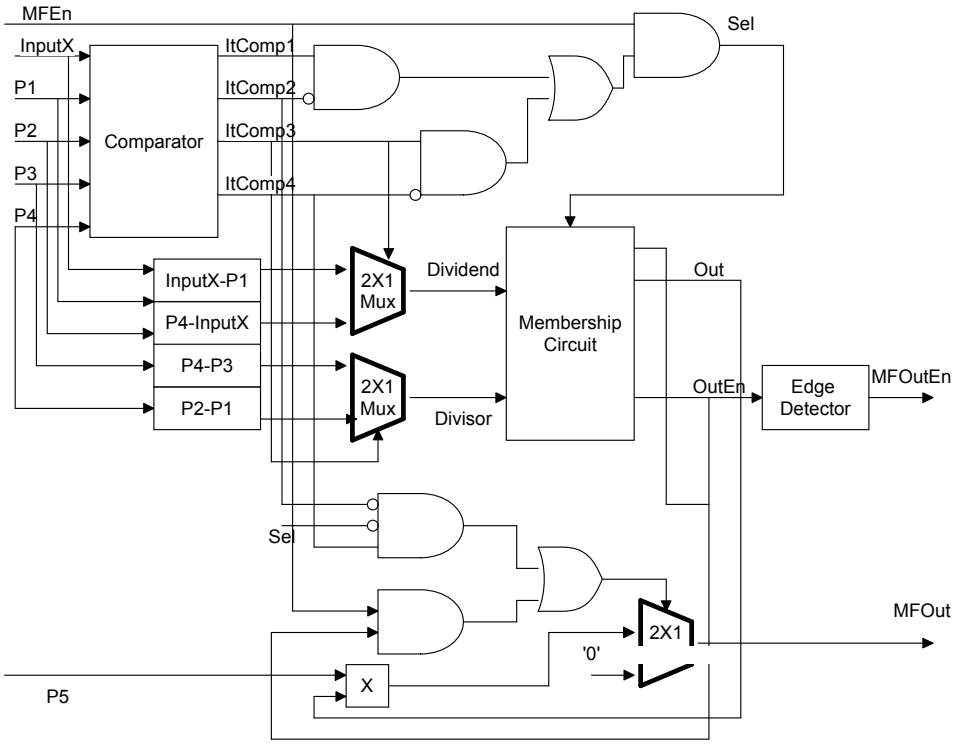


Figure 4.8: Circuit Model of Lower Membership Function

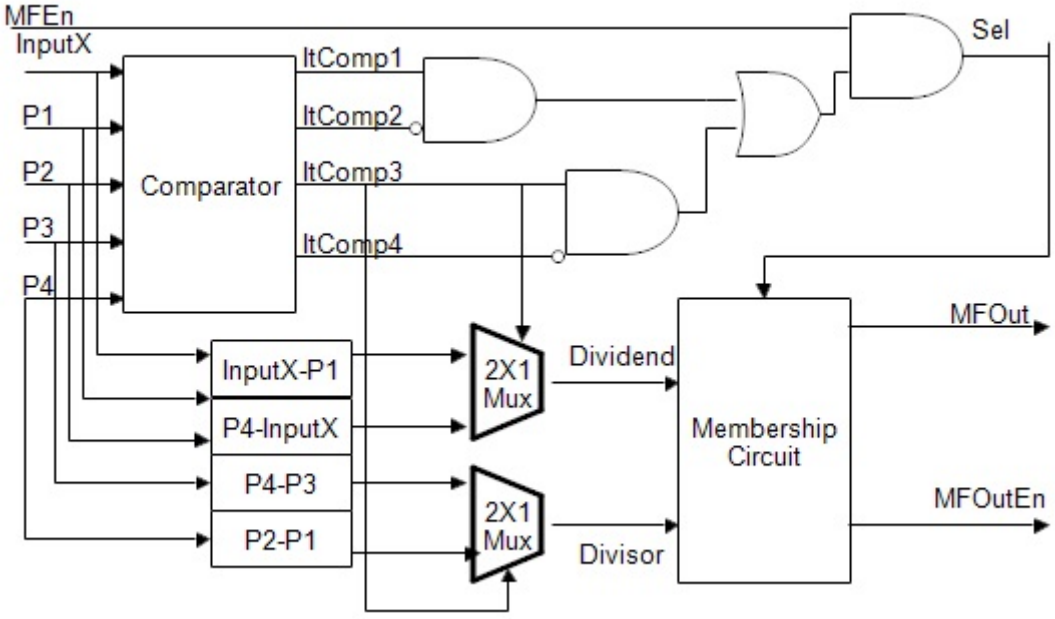


Figure 4.9: Circuit Model of Upper Membership Function

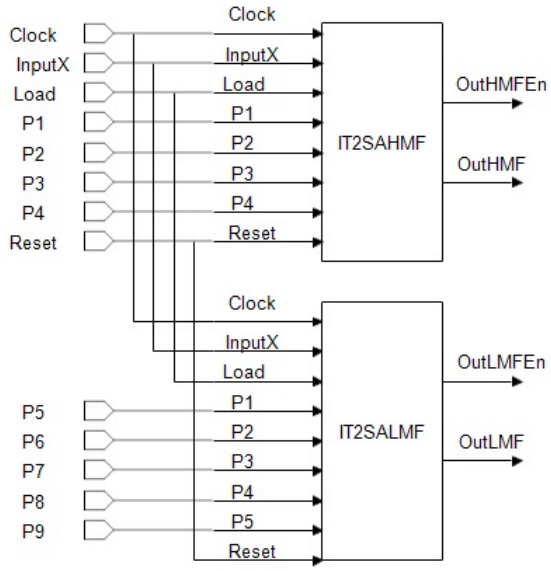


Figure 4.10: Type 2 Fuzzifer Block

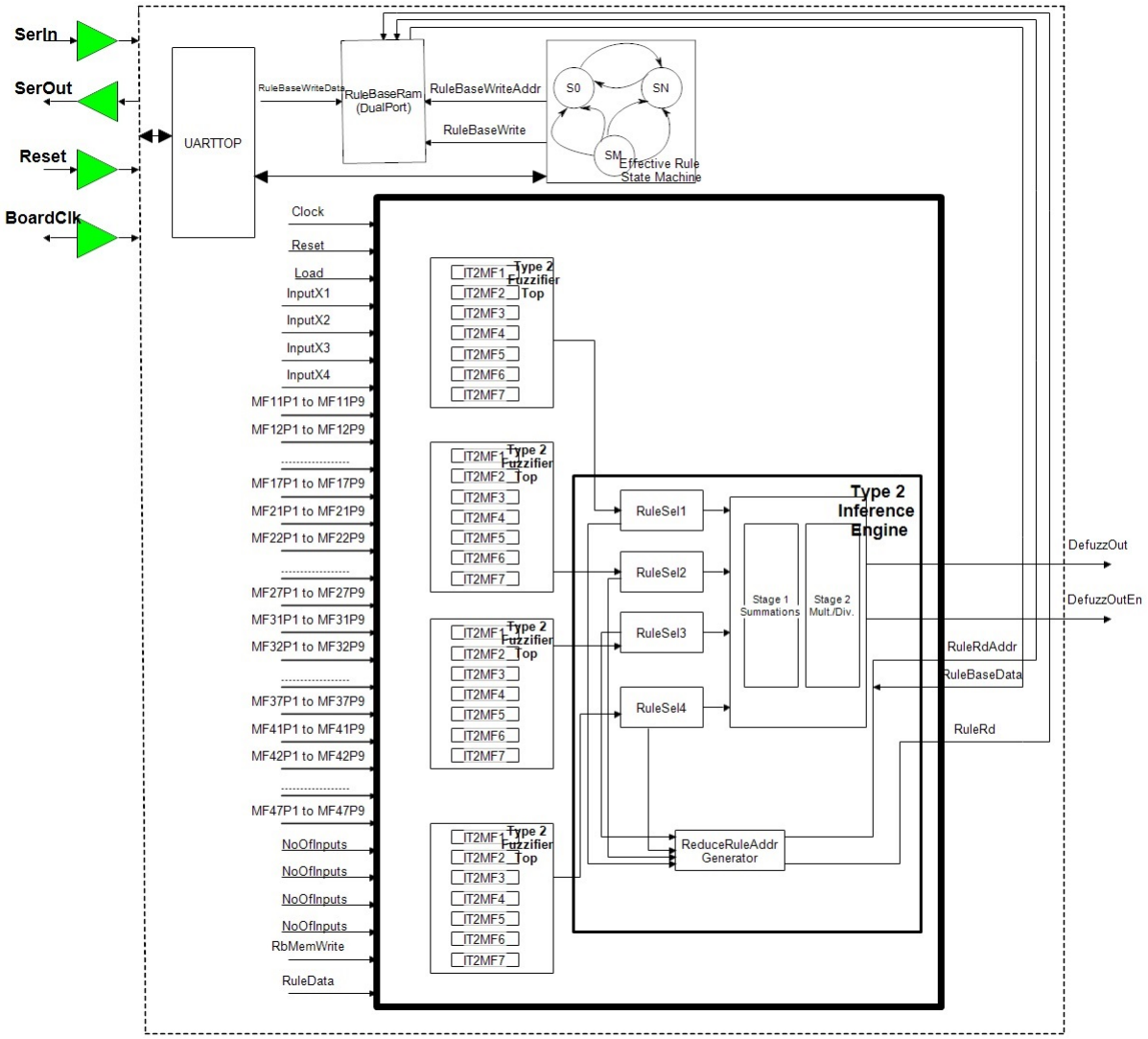


Figure 4.11: Top Level Architecture of Type 2 FLC

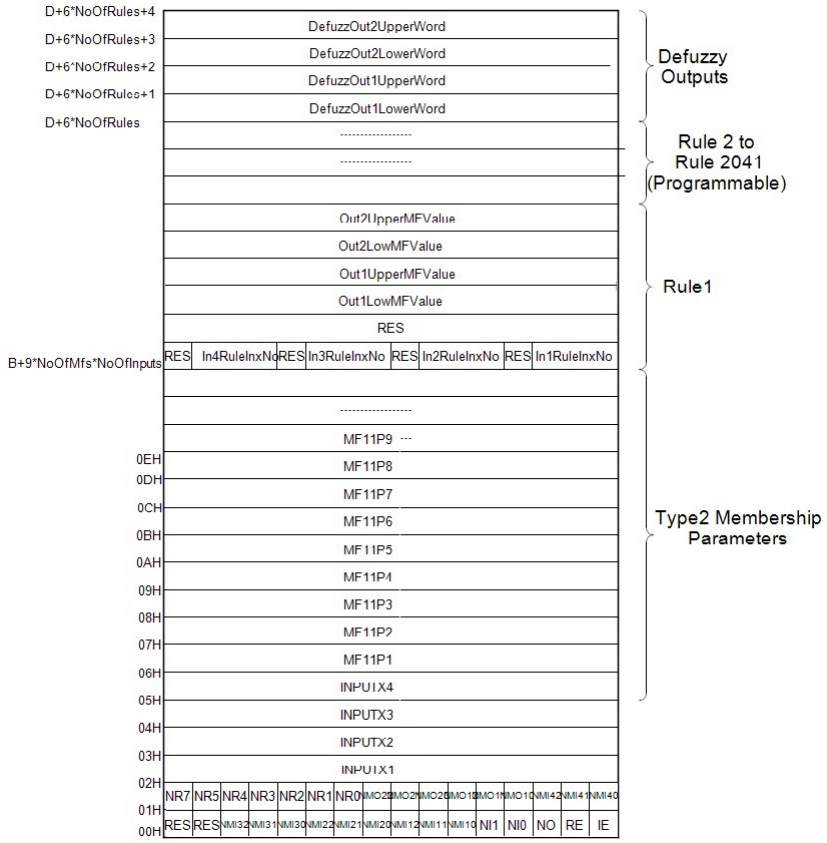


Figure 4.12: Type 2 FLC Memory Map to support different user configurations

fuzzifier is connected four times (one time to each of four inputs) to support four inputs, and its design details were presented in section 4.3. The proposed finite state machine in section 3.4.1 is used to act as control unit as well as to support partial rules. The rule driven circuit discussed in section 2.5 is reused in 'ReduceRuleAddress Generator' module to reduce rules from the methods MRA-2OMF, MRA-3OMF and MRA-4OMF methods based on users choice. All parameters received from GUI are extracted and stored into the memory map of Figure 4.12 as its internal registers. This memory map supports 16-bit variable and provision has been made for the user by parameterizing the data and address widths. The depth of memory in Type 2 FLC ranges from 44 bytes to 27994 (Approx. 28K) bytes. The values here are calculated by taking one input, one rule, one output configuration to four inputs, 2401 rules, and two output configurations. Here, the process of Type 2 FLC starts its operation by configuring Input Enable (IE) and Rule Enable (RE) signals.

4.4.2 Tunable Parameters

From Figure 4.12 for a 16-bit depth memory map the 01H to (D + 6 * No of Rules + 4)H address location (Here 'D' is the variable, which ranges from 0EH to 16EH for different input configurations) is set for fuzzy parameters where,

a. The parameter NO (Number of Outputs) is a single bit value to support maximum configurable outputs as 2. Value '0' for a single output and '1' for two outputs.

b. The parameter NI (Number of Inputs) is a 2-bit value to support maximum configurable inputs as 4. Value '00' is used for single input, '01' for two inputs, '10' for three inputs and '11' for four inputs.

c. The parameter NMI1, NMI2, NMI3, and NMI4 (Number of Membership Functions) relates to a number of membership functions for input 1, input 2, input 3, and input 4. NMO1 and NMO2 relate to a number of membership functions at the output for output 1 and output 2. These parameters are of 3-bit value to provide maximum configurable MFs of 7 to support full four inputs two output system. Value '000' is used for single MF, '001' for 2 MFs, '010' for 3 MFs,'011 'for 4 MFs, '100' for 5 MFs, '101 ' for 6 MFs, and '110' for 7 MFs are used.

d. The parameter NR (Number of Rules) is an 8-bit value to support maximum 256 configurable rules.

e. The parameters INPUTX1, INPUTX2, INPUTX3, and INPUTX4 are crisp input data.

f. The parameter MFXYP1, MFXYP2, MFXYP3, MFXYP4, MFXYP5, MFXYP6, MFXYP7, MFXYP8, and MFXYP9 are Type 2 fuzzy points. Where, X denotes the Membership index number (1 to 7), and Y (1 to 4) indicates the corresponding input number.

g. Each rule consumes six words with the corresponding index numbers of consequents and type 2 antecedents in the fuzzy rule base.

h. Wu-Mendel based output processing unit provide 32 bit value for y_l and y_r .

4.4.3 Inference Engine and Type Reducer

The implementation of inference mechanism is simplified by reading rule base data from rule base RAM, with reduced rule addresses and places them in y_l and y_r array registers. A set of minimum blocks is used to find the effective consequent parts in x_l and x_r registers. The array sizes of these registers are 4, 8, and 16 for number of inputs 2, 3, and 4 respectively.

The Wu-Mendel closed form method [168] finds the bound sets for the interval type 2 fuzzy logic controller. The calculation was performed with closed form expressions. Here, closed form suggests there is a limit or one can say a finite number of steps for the calculation. Unlike KM algorithm, which keeps on iterating until it gets the switch points, here, the outputs are bound set, and these are not the type reduced sets as KM method. The bound set gives the footprint of uncertainty for that particular output. Finally KM's method gets y_l and y_r and average of these two provides the defuzzified value y. But WM method, provides a range for y_l as [$\underline{y_l}$, $\bar{y_l}$] and a range for y_r as [$\underline{y_r}$, $\bar{y_r}$]. The physical significance of these intervals is that they provide the region in which probability of finding y_l and y_r is maximum. Taking advantage of these, the average of these the left and right foot point of uncertainty $\underline{y_l}$ and $\bar{y_l}$ for y_l , $\underline{y_r}$ and $\bar{y_r}$ for y_r can approximate y_l and y_r. These approximate values can provide satisfactory results while saving hardware resource in terms of time and silicon space.

Mini-max uncertainty bounds can express the Wu-Mendel Algorithm [168]. These uncertainty bounds are defined as,

Step1:

$$y_l^{(0)}(x) = \frac{\sum_{i=1}^{N} \underline{x}^i y_l^i}{\sum_{i=1}^{N} \underline{x}^i} \tag{4.7}$$

$$y_l^{(m)}(x) = \frac{\sum_{i=1}^{N} \bar{x}^i y_l^i}{\sum_{i=1}^{N} \bar{x}^i} \tag{4.8}$$

$$y_r^{(0)}(x) = \frac{\sum_{i=1}^{N} \underline{x}^i y_r^i}{\sum_{i=1}^{N} \underline{x}^i} \tag{4.9}$$

$$y_r^{(m)}(x) = \frac{\sum_{i=1}^{N} \bar{x}^i y_r^i}{\sum_{i=1}^{N} \bar{x}^i} \tag{4.10}$$

Step2:

$$\bar{y_l}(x) = min(y_l^{(0)}(x), y_l^{(m)}(x)) \tag{4.11}$$

$$\underline{y_r}(x) = min(y_r^{(0)}(x), y_r^{(m)}(x)) \tag{4.12}$$

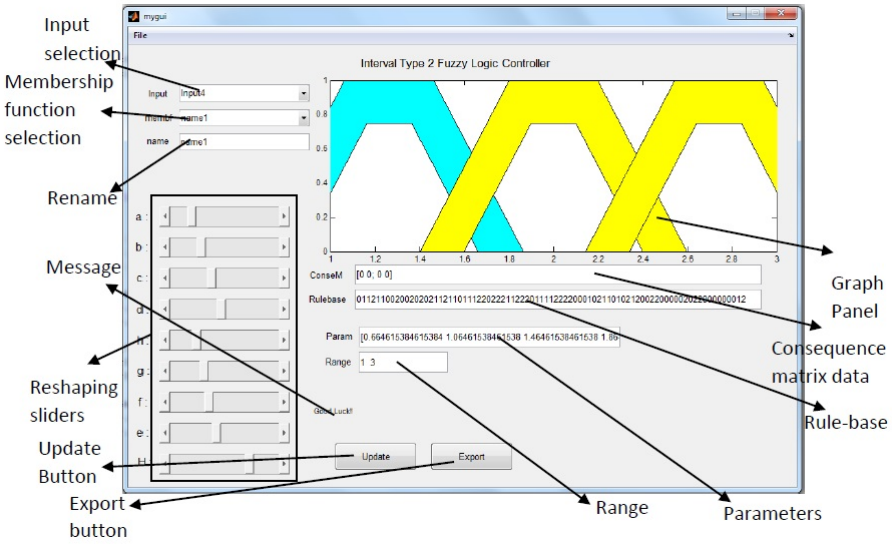


Figure 4.13: MATLAB GUI to configure hardware Type 2 FLC

Step3:

$$\underline{y_l}(x) = \bar{y_l}(x) - \left[\frac{\sum_{i=1}^{N}(\bar{x^i} - \underline{x^i})}{\sum_{i=1}^{N}\bar{x^i}\sum_{i=1}^{N}\underline{x^i}} \times \frac{\sum_{i=1}^{N}\underline{x^i}(y_l^i - y_l^1)\sum_{i=1}^{N}\bar{x^i}(y_l^N - y_l^i)}{\sum_{i=1}^{N}\underline{x^i}(y_l^i - y_l^1) + \sum_{i=1}^{N}\bar{x^i}(y_l^N - y_l^i)}\right] \tag{4.13}$$

$$\bar{y_r}(x) = \underline{y_r}(x) - \left[\frac{\sum_{i=1}^{N}(\bar{x^i} - \underline{x^i})}{\sum_{i=1}^{N}\bar{x^i}\sum_{i=1}^{N}\underline{x^i}} \times \frac{\sum_{i=1}^{N}\bar{x^i}(y_r^i - y_r^1)\sum_{i=1}^{N}\underline{x^i}(y_r^N - y_r^i)}{\sum_{i=1}^{N}\bar{x^i}(y_r^i - y_r^1) + \sum_{i=1}^{N}\underline{x^i}(y_l^N - y_l^i)}\right] \tag{4.14}$$

Step4:

$$\left[\bar{y_l}(x), \underline{y_r}(x)\right], \text{Inner Bounded Set} \tag{4.15}$$

$$\left[\underline{y_l}(x), \bar{y_r}(x)\right], \text{Outer Bounded Set} \tag{4.16}$$

$$y_l(x) \cong \frac{\bar{y_l}(x) + \underline{y_l}(x)}{2} \tag{4.17}$$

$$y_r(x) \cong \frac{\bar{y_r}(x) + \underline{y_r}(x)}{2} \tag{4.18}$$

$$y(x) = \frac{y_l(x) + y_r(x)}{2} \tag{4.19}$$

4.5 Results and Discussion

In this section, the proposed type 2 fuzzy system based on successive approximation interval type 2 fuzzifier is compared to existing type 2 hardware IT2FLCs for performance analysis.

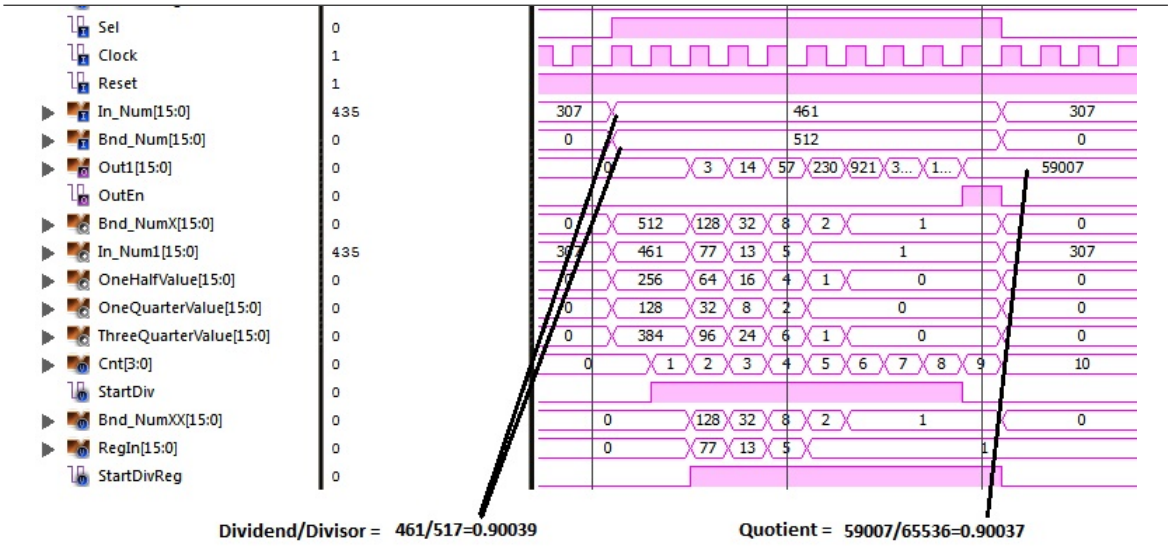


Figure 4.14: Functional simulation result of Successive Approximation Division method

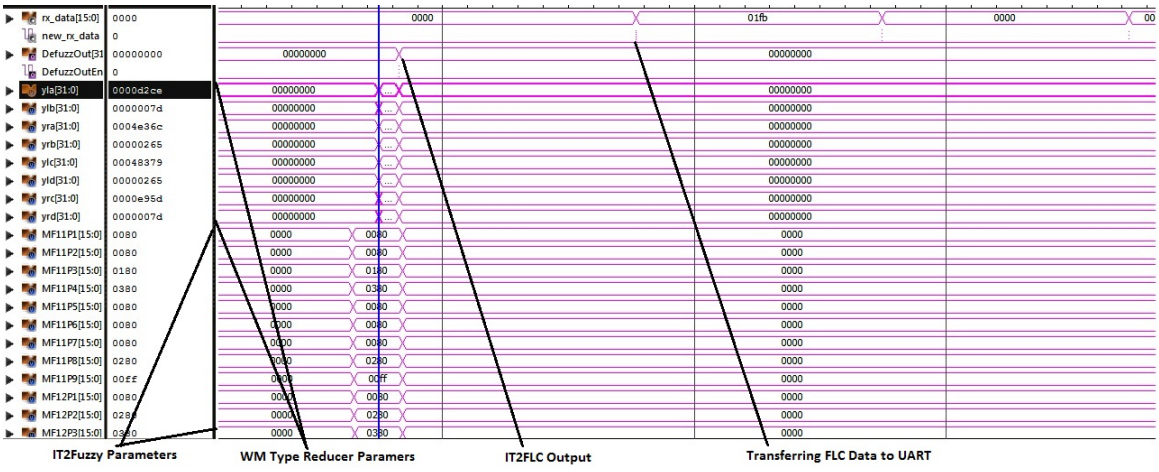


Figure 4.15: Functional simulation result of SAIT2FLC

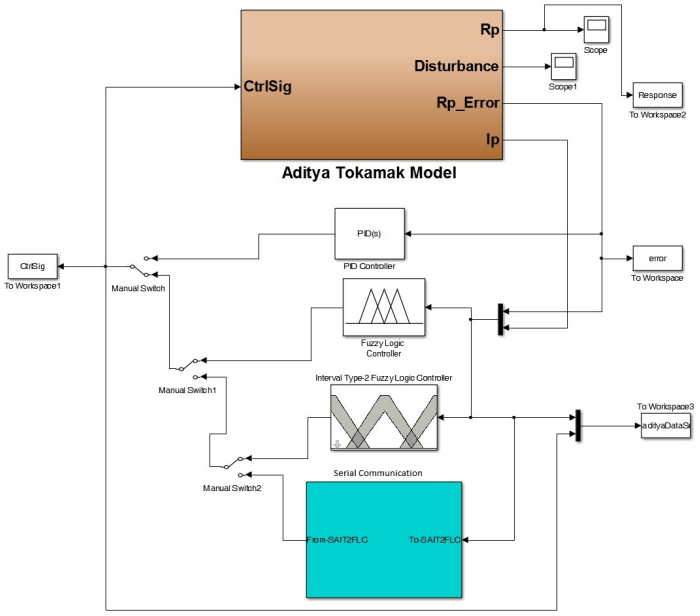
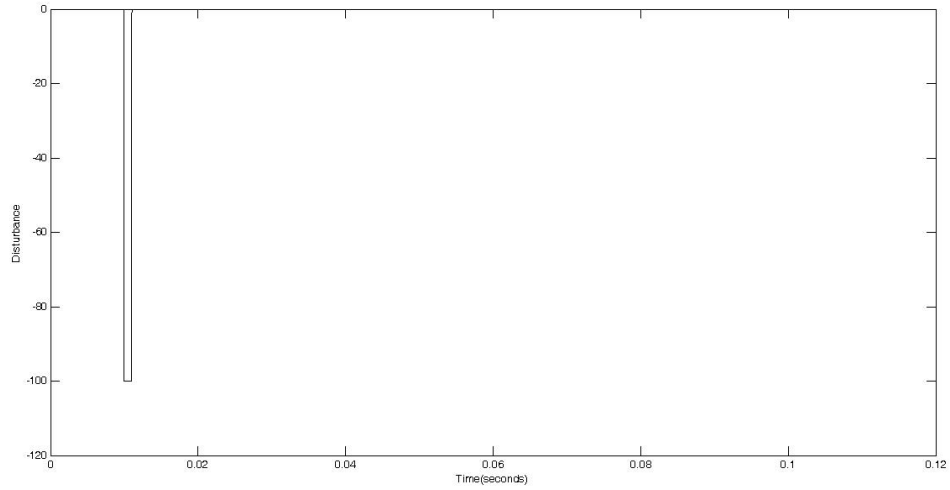


Figure 4.16: Simulink model of radial plasma position control in Aditya TFTR with FLC and SAIT2FLC

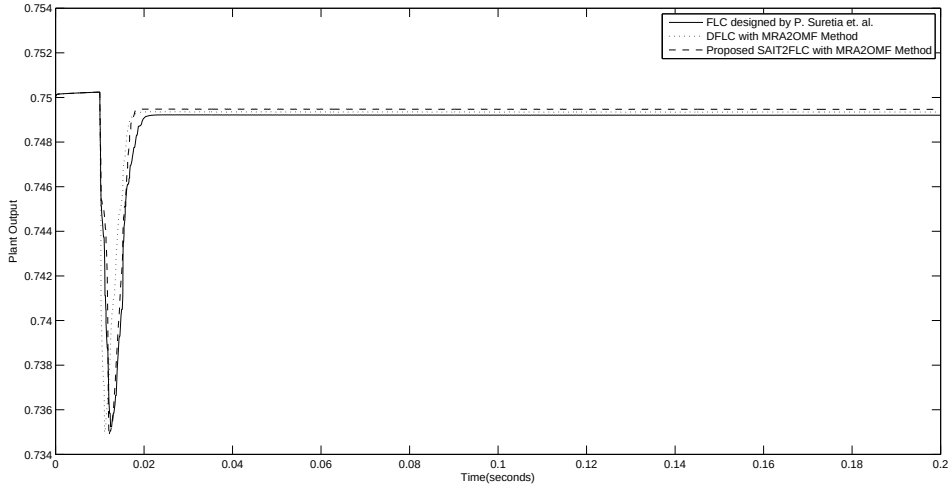
Table 4.1: Comparison of performance parameters of FLC [1], and DFLC with MRA2-OMF, SAIT2FLC with MRA2-OMF,MRA3-OMF,MRA4-OMF

Parameters	FLC [1]	DFLC (MRA-2-OMF)	SAIT2FLC (MRA-2-OMF)	SAIT2FLC (MRA-3-OMF)	SAIT2FLC (MRA-4-OMF)
Rise Time	0.0025	0.0023	0.0021	0.0019	0.0018
Settling Time	NaN	0.0249	0.0177	0.0169	0.0162
Overshoot	0	0	0	0	0
Undershoot	0	0	0	0	0
Peak	0.7483	0.7492	0.7493	0. 7493	0.7494
Peak Time	0.02	0.0235	0.018	0.017	0.016

The tunability of this FLC is evaluated by testing with different test vectors using MATLAB GUI presented in Figure 4.13. Figure 4.14 shows the functional simulation of successive approximation division with the latency of 9 clocks. The latency and cycle time for various hardware implementations of Type 2 FLC is presented in Table 4.3. From the analysis it is seen that the proposed method works efficiently with less latency. Figure 4.15 illustrates the functional simulation of Type 2 FLC output on UART SEROUT port for external interface reading. An experiment was conducted for evaluating the performance of this FLC using the radial position control of a plasma column in Aditya TFTR. Its system modelling and simulink model were discussed in section 3.7. UART is used for serial communication to establish data exchange between SAIT2FLC and Simulink. Using Manual switches, simulation of all controllers were carried out sequentially as shown in Figure 4.16. A comparison of control performance of the proposed SAIT2FLC with MRA2-OMF method



(a) Input Disturbance Signal



(b) System Response

Figure 4.17: Performance of various controllers in presence of disturbances in plasma position

and DFLC with MRA2-OMF proposed in section 2.5 is plotted in 4.17 and their control parameters are tabulated in Table 4.1. In Table 4.2, the logic utilization and latency comparison of two methods are shown. From these tables, it can be observed that SAIT2FLC hardware has an improvement in control parameters with 8% faster rise time and 28% faster settling time at the cost of chip area and latency compared to DFLC. SAIT2FLC consumes more DSP48E slices to support multiply accumulation operation in type 2 output processing. This architecture also consuming more clocks to finish type reducer task resulting 37% slower speed in FLIPS when comparing with DFLC MRA2-OMF design.

Table 4.2: Hardware Implementation: Comparison of proposed method SAIT2FLC with DFLC

Proposed Methods	BRAM utilized		DSP48Es	LUTs	Cycle Time	Latency
	32K	18K				
DFLC with MRA2-OMF	0	1	2	2327	6.608 ns	99.12 ns
DFLC with MRA3-OMF	0	1	3	2551	6.754 ns	101.31 ns
DFLC with MRA4-OMF	0	1	3	2618	6.913 ns	103.695 ns
SAIT2FLC with MRA2-OMF	1	1	25	8724	8.129 ns	267.993 ns
SAIT2FLC with MRA3-OMF	1	1	28	9413	8.513 ns	280.652 ns
SAIT2FLC with MRA4-OMF	1	1	28	9881	8.605 ns	283.685 ns

Table 4.3: Performance of Successive Approximation Based Type2 FLC with other methods

Year	Type 2 Fuzzifier Method	Cycle Time	Latency
2003	G. Louverdis et al. [169]	15 ns	1665 ns
2004	Melgarejo et al. [170]	29.789 ns	1026 ns
2014	Schrieber et al. [171]	20 ns	460 ns
*	Proposed SAIT2FLC	8.129 ns	267.993 ns

4.6 Summary

In this Chapter, a FLC based on type 2 fuzzifier with successive approximation technique is presented. In this system, for a 16 bit fuzzy process the fuzzifier took nine clock cycles each with 8.129 ns cycle time and provided latency of 73 ns. This time is very low compared to other division algorithms used in type 2 fuzzifier. The overall speed of SAIT2FLC reached 3.7 MFLIPS. Even though the SAIT2FLC has an improved control performance, the major drawback of this design is the complexity of the design and silicon area. Latency of the system is another drawback as the SAIT2FLC worked with higher latency of 267.993 ns compared to DFLC with 99.12 ns for ADITYA TFTR problem. The next chapter deals with the rule base optimization with Genetic Algorithm (GA) to reduce the complexity and latency of DFLC on FPGA platform.

Chapter 5

FPGA Implementation of Genetic Algorithm (GA) based Rule Optimized Fuzzy Logic Controller

Preface

In this Chapter, a self-tuned rule-optimized Multi-Input and Multi-Output (MIMO) FLCs is implemented on FPGA. The design of membership functions in the rule base is made with the aid of Genetic Algorithm (GA). Flexibility in FPGA design is implemented through tuning of FLC parameters. The system is modularized as rule base development, rule base transfer and computations on FPGA. Based on the system, an experimental dataset is obtained, which is utilized in a capable computing platform so as to develop a fine-tuned fuzzy rule base. The synthesized rule base is transferred to FPGA along with the user-provided inputs through a GUI. The GUI also displays the output result sent by FPGA. The communication between the GUI and the FPGA is achieved via UART. The proposed FLC is implemented on Xilinx Virtex-5 LX110T board. This dedicated single chip architecture performs high-speed fuzzy inferences with processing speed up to 9.88 MFLIPS at a clock frequency of 247MHz using eight rules for two input variables with 16-bit resolution. Experiments of software implementation and hardware software co-design implementation shows encouraging result.

5.1 Introduction

Fuzzy logic controllers (FLC) have immensely contributed to the industrial sector as a powerful tool to solve complex and non-linear control problems. Fuzzy logic, being the mathematical emulation of human reasoning, are developed by human intelligence, which have helped in designing intelligent control systems with advanced features in handling environmental changes and system level faults.

As discussed earlier in the thesis, FLC are designed by taking crisp inputs from the user, and they perform complex mathematical computations to provide the output. The entire design constitutes of mathematical computations, which are difficult to synthesize practically. As they have potential for large error even with the best of the design implementations. The classical theory of fuzzy logic has failed to develop the systems thoroughly and efficiently for increased number of variables, conditions and become less powerful with the requirement of MIMO systems. To avoid any such problems in the design systems having accessible user interfaces with the availability of input-output data, intelligent mechanisms to work out the rule base can be used. Some of the design methodology include neural networks, regression, evolutionary algorithms, etc. [172–174]. In this Chapter, fuzzy systems are first trained with known input-output results and then tested for new input conditions.

A number of FLCs have been designed with intelligent techniques [175–178] for different application. But the designing of these controllers requires a thorough knowledge of the controlled process. FLCs are designed based on human intelligence, i.e. by the experts. Most of these processes are non-linear and depend on a large number of parameters, which results in the rigorous mathematical representation of the process. It is tough to incorporate each of the parameters while designing the FLC. This Chapter discusses methods to create optimized fine-tuned rules to an FLC on hardware. The designed fuzzy system has potential to reduce the level of complexity in terms of chip area and speed.

This Chapter elaborates designing of an optimized FLC using Genetic Algorithm (GA). The FLC proposed extracts tuned rule base automatically by analyzing the training data set alone, making the design superior where,

i. Human knowledge brimming with possibilities of error is relied upon to make the decision-making process.

ii. Enough prior knowledge is required for decision making.

iii. Many conditions are to be checked to design the system, which cannot be done manually, hence prone to error.

iv. Solution to the system must be designed with transparency to non-experts.

FLC design is normally an offline process, so the system is tuned before use. Some of the techniques used to train FLC are discussed here: H. Nomura [179] reported a self-tuning method for fuzzy inference rules, employing a descent method for TS fuzzy rules with constant outputs and isosceles triangular MFs. Y. Glorennec [180] presented an adaptive controller using fuzzy logic and connectionist methods. Whereas, Siarry et al. [181] used the gradient descendant method for optimizing TS rules with symmetric and asymmetric triangular membership functions (MFs) proposing the 'centered TS rules' for avoiding a particular class of local minima. O. Cordon et al. [182] used real coded GA with some genetic operators for tuning the membership points. Similarly GA for designing an adaptive FLC has been attempted by many researchers [183–185]. These works analyzed several attributes for the rule base generation for FLC development, but the number of rules increases exponentially with the increase in the number of considered attributes. For example, in a 2-input system, if the number of attributes per input is 4, then total number of rules will be $4^2 = 16$, For 3-input and 4 attributes system, it will be $4^3 = 64$. So with increased number of inputs and corresponding attributes the number of rules becomes very high, which results in computational complexity and memory complexity. But in the present designing principle, it is proposed to reduce rules with required efficiency level. The facility of FLCs to capture the automatic learning from data and render it into a rich control strategy without the need of mathematical model of the system or expert knowledge is the key advantage of the proposed design.

The mathematical model of the system under control has led to a significant increase in the number of control applications in the last fifteen years [172, 186]. This has also propelled the development of different approaches to implementing fuzzy inference systems. These strategies range from completely software or hardware solutions or mix of two. This chapter considers a hybrid realization, which allows fair trade-off between flexibility and inference speed [187, 188]. Hybrid strategies require software task execution and fixed hardware to execute complex, time-consuming tasks, usually the Fuzzy Inference Process (FIP) [189].

Many hardware solutions choose FPGAs for their advantages discussed in section 2.1. Chekired *et. al.* [190] implemented FLCs on FPGA, where they used fixed rule base structure and hence give lower FLIPS (Fuzzy Logic Inferences Per Second).A. Messai *et. al.* [191] developed genetic algorithm based FPGA design of Maximum power point tracking system. Here, the design is optimized with 25 rules for 2-input 1-output system, and the design operated at a maximum frequency of 97 MHz, which eventually resulted in to high cycle time or low FLIPS. The proposed generalized FLC with optimized rule base implementation on FPGA is well suited to the applications, where cycle time of the system is very less.

Generally, software level design has advantage over hardware level design for validating multiple conditions, handling add-in functionality of the system and easy of configurability. Hardware typically provide faster functionally superior to software owing to processing of high volume of data. With current technology the software can be embedded in memory circuits, thereby providing hardware-software co-design. Through in this type of design principle, the overall system has higher performance of hardware also provides smooth designing and reprogrammable capability of software. This co-design can be made possible with the development of high-performance processor core, embedded memory circuits and faster communication technologies between hardware and software.

Hardware/software co-design principle for the fuzzy system designing has been discussed here. The chapter also includes hardware direct interaction with the user interface (PC), communication and computing in FPGA along with serial communication through UART.

5.2 System Architecture

The proposed system architecture for GA based FLC on FPGA using software- hardware co-simulation is presented in Figure 5.1. The training data set is prepared from the experimental observations of the system is provided to the computer. The training data sheet is used to extract the initial rule base and then optimized it using GA. Accuracy of the rule base depends on the distribution of data points in the data sheet over the total system range of application.

Once the rule base is designed, the parameters of the rules, which include input variables and control information are sent through serial communication protocol like RS-232 of the computer. The UART module on the Xilinx board receives the rule base and saved in the

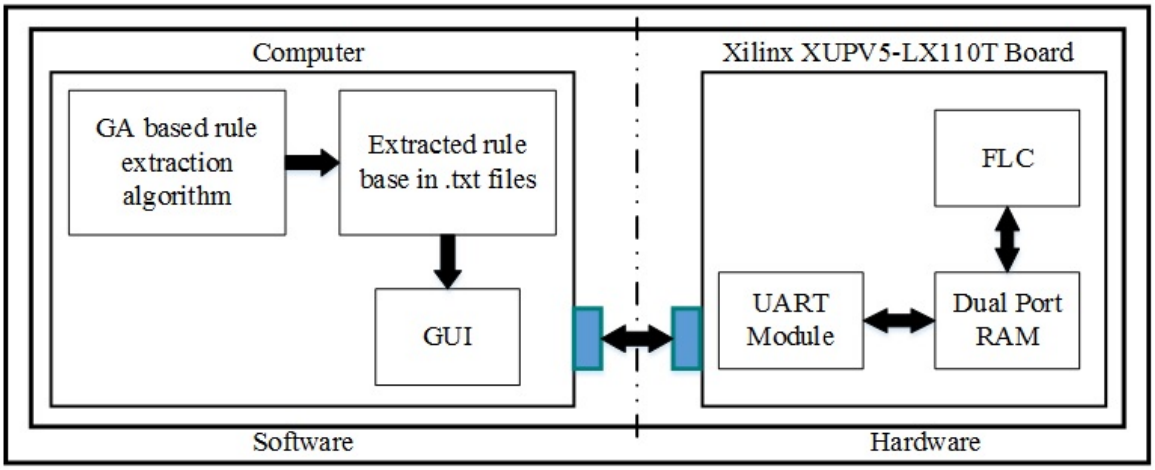


Figure 5.1: System Architecture of GA-FLC on FPGA

dual port block RAM on the FPGA through its Port A. This completes the initialization of the rule base at the hardware level. Following this for testing purpose input parameters of a data point are taken through a GUI to the computer and are sent serially to the Xilinx board. All the rules, input values, control information, are read from port B of the dual port RAM, which initializes the fuzzy inference processing hardware to receive the desired crisp output. This value is again transferred serially back to the computer for display on the GUI. The memory address map and control information of dual port RAM is shown in Table 5.1. All the parameters here are considered in fixed point notation Q8.8 for simplicity in hardware implementation.

In the hardware, the core is divided into two sections (a) software section and (b) hardware section. The software section is executed in the Computer (PC) for the designing and tuning of the rule base and the hardware section is implemented on the FLC executed on the XUPV5LX110T board in real-time. The specification of the FLC proposed to be built in this work is as follows:

a. No of Inputs : 4
b. No of Outputs: 2
c. Shape of Membership Function: Triangle
d. Number of fuzzy sets per input and output variables: 7
e. Resolution of membership values: 16 bit
f. Implication Model: Mamdani
g. Aggregation Model: Mamdani

h. No of rules field programmable

i. Configurable on the field by GUI application

The entire system design can be subdivided into following stages:

- Rule base extraction using GA
- GA optimized rule transmission through UART to FPGA
- Hardware architecture of FLC in Virtex5 LX110T FPGA.

All the above stages of design are discussed in the following sections.

Table 5.1: Memory Space

Address	Access	Name	Description
000H-001H	Read/Write	CTRLREG	Control register for software and hardware updation
001H-221H	Read	RULEBASE	Based on the number of rules, the address range is varied within this address.
222H-225H	Read/Write	INPUTVAL	Address to store input variables in 16 bits
226H-22AH	Write	CRISPOUT	To store Final Crisp output

5.3 Rule Base Extraction

The most important step in designing an FLC involves rule base extraction. The rules in a rule-base can be optimized either using expert's knowledge of the process or by using the available experimental dataset of the process. The later approach automatically learns the process attributes without relying on the expert's thorough process understanding.

Rule extraction is achieved through algorithms like Fuzzy C-Means (FCM) [192, 193], Hard C-Means (HCM) [194, 195], K-Means algorithms [196, 197] etc. In this work, K-Means clustering algorithm to design the initial rule base is used. The number of clusters equals the number of rules in the rule base. After fixing the number of rules as per the requirement, clustering is applied to generate rules out of the dataset at each run.

Following this process, GA is used to optimize the initial rule base by tuning the centers and boundaries of each rule. Through several epochs GA evolves to provide more accurate rule base. The tuning parameters of the GA can be set as per the user's desired accuracy level in accordance to system's memory and time complexity.

5.3.1 Rule Base Initialization

Initially a number of rule bases equal to the population size of GA is designed. The number of rules is same for each rule base. K-Means clustering algorithm is then applied to design each rule base. In order to initialize one rule base 'c ' number of data points are chosen randomly so as to accommodate flexibility in the designing. They are chosen as the initial cluster centers for each of the clusters. Then, the Euclidean distance of each data point in the dataset is calculated from each of the cluster centers. The data point is included into a cluster whose center has the minimum distance from the data point. Thus, all the data points are distributed among different clusters.

In each of the clusters, the feature wise mean is taken for all the data points belonging to that cluster, and these mean values are set as the new cluster centers for the particular cluster. The process is repeated taking the modified cluster centers as the new rule base. This process is continued till the difference between all the points in the current rule base and the former rule base remain below a pre-defined threshold. As two consecutive rule bases differ by a small margin, as set by the threshold, the convergence of the rule bases is ensured. Thus, a final clustered rule base is generated.

Triangular membership function is chosen as the MF for the designed algorithm. Each of the points in the rule base forms the center of the corresponding triangular MF. In each of the final clusters, the feature wise minimum and maximum are selected, which are set as the two endpoints for the corresponding triangular MF for that particular rule. Let,

C=Number of rules to be designed with a rule base, and $1 \leq i \leq c$,

n= Number of data points in the data set, and $1 \leq k \leq n$,

m= Number of features in each data point, and $1 \leq j \leq m$,

D_k=k^{th} data point in the data set = $\lfloor D_{k,j} \rfloor = \lfloor D_{k,1}, D_{k,2} \ldots D_{k,m} \rfloor$

R_i=i^{th} rule in the data set = $\lfloor R_{i,j} \rfloor = \lfloor R_{i,1}, R_{i,2} \ldots R_{i,m} \rfloor$

C_i=i^{th} cluster,

$d_{k,i}$ = distance of D_k from R_i, i.e. the cluster center C_i

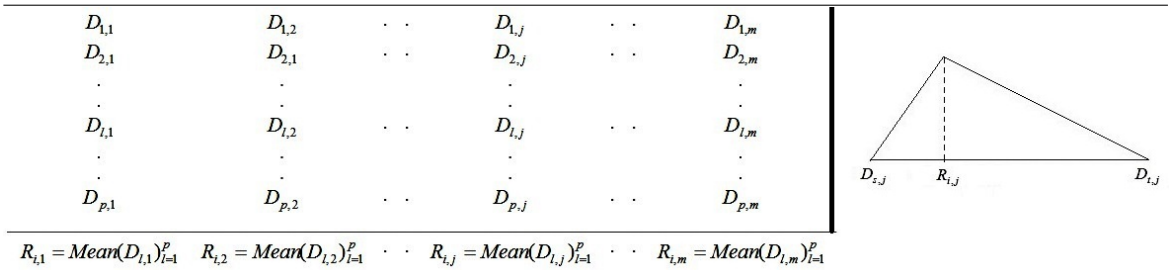


Figure 5.2: Rule Base Design

Then,

$$D_k \in C_i \Leftrightarrow \min_{i=1}^{c} \left\{ \sqrt{\sum_{j=1}^{m} (D_{k,j} - R_{i,j})^2} \right\} = d_{k,i} \tag{5.1}$$

After clustering, if the number of data point in cluster C_i is p. Then rule R_i can be constructed from C_i by taking the feature wise mean of all data points $D_k \in C_i$, as shown in Figure 5.2.

5.3.2 A Genetic Algorithm for Tuning of the Rule Base

Real-coded GA has been used for optimization. The GA has been designed to deal with the situation where, the antecedents and the consequent all are membership functions. The proposed algorithm has been used for triangular MFs, but it can also be extended to any other type of MF. The parameters of GA are set as per the design requirements. Figure 5.3 presents the flowchart for GA algorithm used in this work. The steps of GA are explained below.

5.3.2.1 Step1: Selection

As per the algorithm used in this work the population size has to be even. The clustering algorithm and initial rule base designing method explained above has been used to generate initial rule bases for each population. Thus, each population corresponds to one rule base for which, the total squared error can be calculated for all the data points in the training data sheet. Here, the objective is to reduce the sum of squared errors.

5.3.2.2 Step 2. Crossover

Sum of squared errors is calculated for each rule base. These are sorted in ascending order, so the 1^{st} one corresponds to the best rule base, i.e. rule base with the minimum squared error and the last one with the maximum squared error. The Roulette Wheel technique

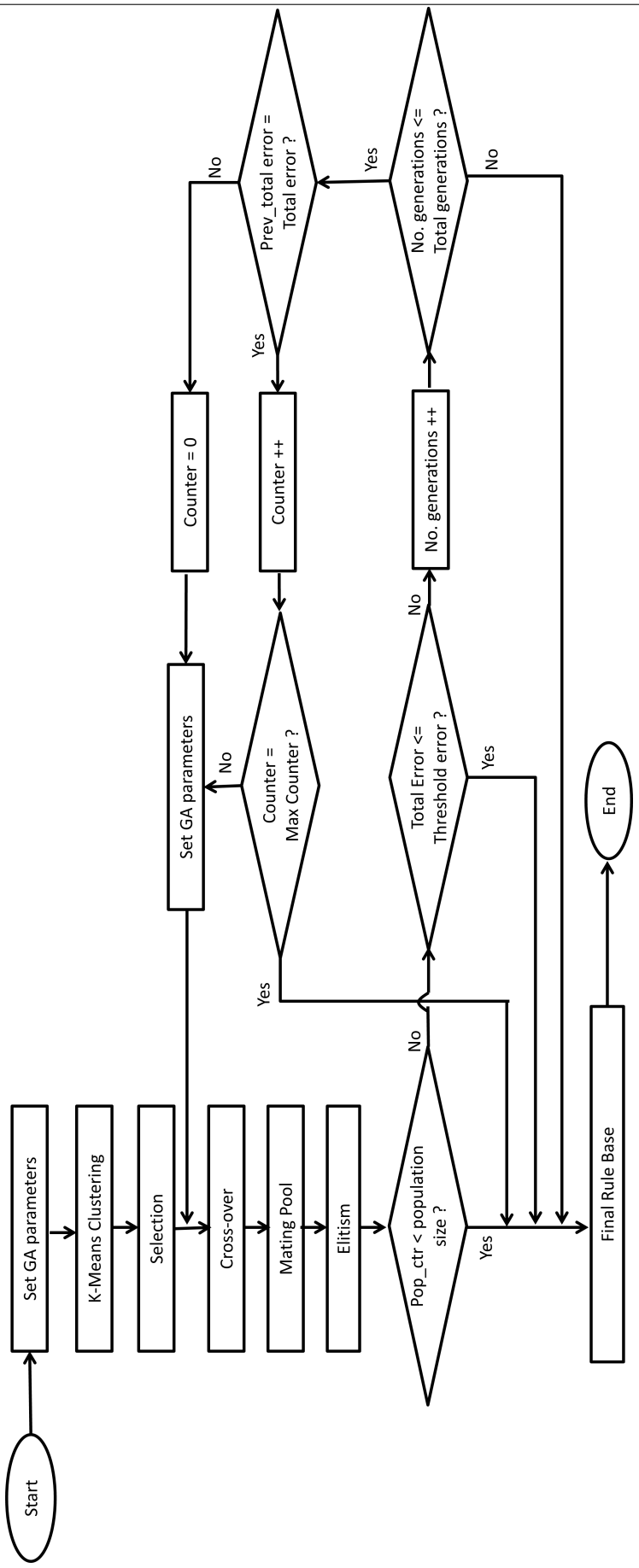


Figure 5.3: Flow diagram showing the GA based optimization of the fuzzy rule base

is used to select the populations for crossover purpose. The purpose of a crossover is to generate new children from their parents. In this case, both the parents are rule bases and crossed-over is used to generate child rule bases. A crossover probability is chosen as a design parameter. In this work it is chosen randomly between 0 and 1. If the rule value is less than the crossover probability then crossover is carried out, else no crossover is applied. The concept of crossover is that the better parent gives best offspring. The total number of crossovers is limited to population size/2. During each crossover one parent is from the first half of the population and others from the other half of the population. The corresponding parent with whom crossover takes place is chosen randomly on Roulette wheel method.

After each crossover, two new children are generated. If no crossover is done between the two parents then parent parameters are transferred to their children. Thus, the total number of children produced after crossover of all parents equal the population size. The technique used for crossover is discussed below.

Crossover is carried out between the two populations, i.e. between two rule bases. For triangular MFs, only the peaks of the triangles, which construct the rule base, take part in the crossover process. Suppose the two triangles, which take part in crossover have the base points given by [a_1, m_1, b_1] and [a_2, m_2, b_2], then d_1, d_2 are calculated as below.

$$d_1 = \frac{m_1 - a_1}{b_1 - a_1}, d_2 = \frac{m_2 - a_2}{b_2 - a_2} \tag{5.2}$$

These d_1 and d_2 values were then interchanged between the 2 triangles, i.e. the new value of m_1, m_2 are

$$m_1 = a_1 + d_2 * (b_1 - a_1), m_2 = a_2 + d_1 * (b_2 - a_2) \tag{5.3}$$

The end points of the triangle remain unchanged. The shape of membership functions before and after the crossover is shown in Figure 5.4.

5.3.2.3 Step3: Mutation

Mutation is the process of maintaining genetic diversity from one generation of population to the next. Here, the newly generated triangles are again modified slightly expecting for a better rule base. The designer sets a mutation probability and a mutation fraction as a design parameter. The maximum number of points that can be mutated is equal to mutation probability times the total number of points. In this case, the triangle centers and the boundary points were mutated to result in better rule base. The above algorithm includes

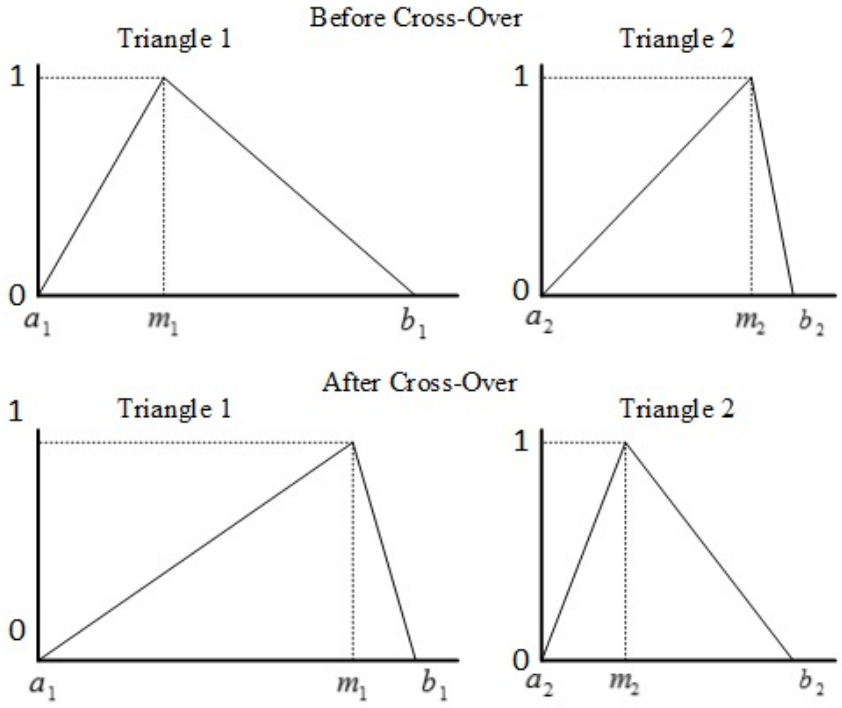


Figure 5.4: Shape of membership functions before and after crossover

that; if the center point of a triangle gets mutated then the boundary points of the triangle will also get mutated providing a new triangle.

A number is chosen randomly in between 0 to 1. If that number is below the mutation probability then mutation is applied, otherwise no mutation occurs. The mutation of a triangle described by the points $[a, m, b]$ is carried out as given below. First the minimum distance is calculated, i.e. $d = min(m - a, b - m)$. Then a new value of 'm 'is assumed in the neighborhood of 'm 'within a given region,

$$([m - d] * mutationfactor, [m + d] * mutationfactor) \tag{5.4}$$

Similarly the boundary points of the triangle also get mutated. a and b are assumed within a region given by,

$$\begin{aligned}&(a - [m - a] * mutationFactor, a + [m - a] * mutationFactor)\\&(b - [b - m] * mutationFactor, b + [b - m] * mutationFactor)\end{aligned} \tag{5.5}$$

The m value used in updating the boundary points is the old value of m before it is modified.

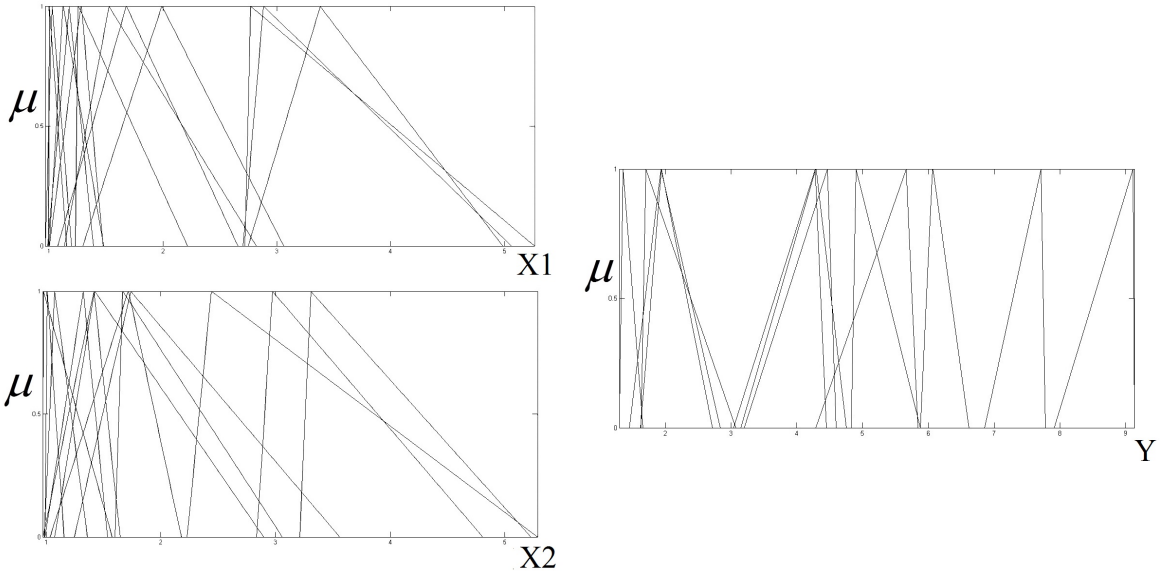


Figure 5.5: Reduced Rule Base with GA optimization

5.3.2.4 Step4: Elitism

The total squared errors for all the populations, i.e. for all the rule bases generated before crossover, after crossover, and after mutation are calculated. The rule bases for the next generation of GA are chosen from the mating pool, based on the minimum total squared error, i.e. all the rule bases shorted in ascending order of their total squared error, and the number of rule bases carried forward to the next generation equals the population size of the GA. These new populations are again modified through the previous procedure in subsequent generations resulting in an optimum rule base with a minimum total squared error. The plot of optimized rule base drawn as in Figure 5.5 for input X1, input X2 and output Y. This generation process terminates if,

- The number of generations set by the designer is completed.
- The total squared error comes below the required error level.
- The total squared error does not improve over a number of generations.

5.4 Rule Base Transmission

After the rule base optimization, the system parameters are sent through the serial communication over the UART to the FPGA and vice versa using the Matlab GUI is presented in Figure 5.6. The transfer of all optimized rules, input values, and control

information follows fixed point representation; the following points explains the data representation and a total number of bytes to finish the communication process.

1. The data to be sent is multiplied by 256 i.e. 8 bit left shift in binary value.
2. The obtained result is rounded to nearest integer value.
3. This value is represented in 16 bits Q8.8, Where first 8 bits constitute integer part and the other bits constitute fractional part format and sent over the UART in chunks of bytes with higher byte followed by lower byte.
4. Initially control registers are transmitted, which indicates the availability of the crisp input values and the number of rules, Also initiates the hardware FLC process.
5. The transmission of data points of reduced rules begins in the reverse direction, i.e. the 1^st upper value of the triangle of the rule is sent. Then the center value of the rule followed by the lower value.
6. Once the first rule is sent over the UART, Following the 2^{nd} rule is sent followed by the 3^{rd} rule. This sequence is followed till the final rule is sent.
7. After the rule base was sent successfully, High byte of n^{th} crisp input is sent, then its low byte. All the crisp input values were sent as per the number of input parameters.
8. Total number of Points = No. Of Rules * No. Of Points required to represent each membership function * (No. Of inputs + No. Of outputs)
9. Total number of bytes to be transferred for parameter initiation is = Total number of points * 2
10. For examples, let us design a system for 2 inputs and 1 output. Triangular membership functions are used to design 8 rules in the rule base both for input and output side. 3 points are required to represent each triangle.
11. Total number of data points to be transferred = 8 *3 * (2 + 1) = 72.
12. Total number of bytes (In Q8.8 format) to be transferred = 72*2 = 144.
13. After 144 bytes have been captured, the FLC treats the next pair of bytes as inputs. That finishes the initiation process.

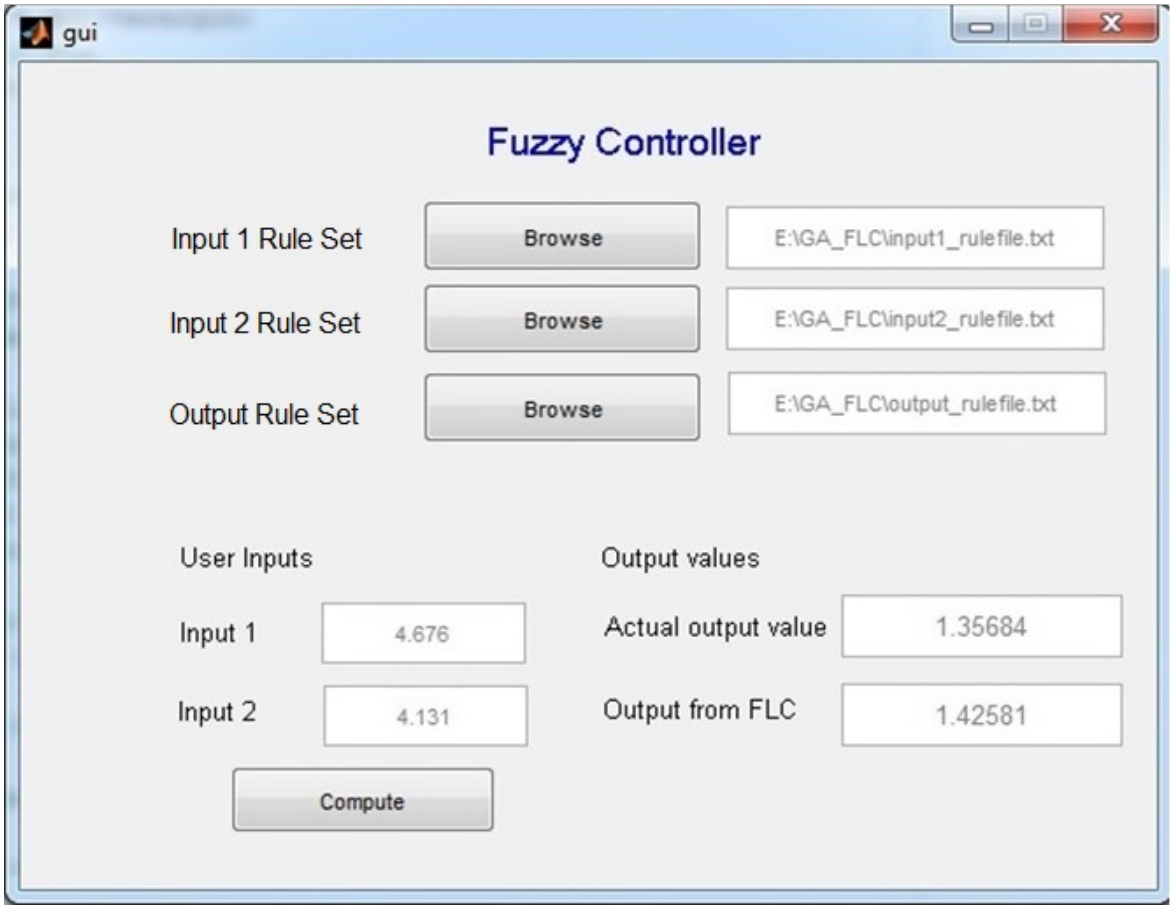


Figure 5.6: Designed GUI for FLC using uses MATLAB for rule transmission to FPGA and computed value received from FPGA

5.5 Hardware Architecture of the FLC

The design goal in this exercise is to enable extraction of synthesized rule base from a computer hardware platform onto a dedicated hardware platform for the fuzzy system. The UART is chosen as a communication interface due to its simplicity in design and results in reduced debugging time. Figure 5.7 shows the hardware block diagram of the fuzzy logic controller module. The choice of components for the architecture is dictated by the terms imposed by the HDL as well as the synthesis software. A word size of 16-bits is considered as the numbers represented in Q8.8 format. The software sends the optimized rule base followed by the triangular membership functions. Since triangular functions are used, a set of three points is required to represent each function. The software transmits these points as a set of two bytes over UART. The membership function points of the rules are stored in RAM locations after the UART on FPGA has captured the bytes. Following the rule base extraction, the crisp input values are sent to hardware and are also stored in RAM locations.

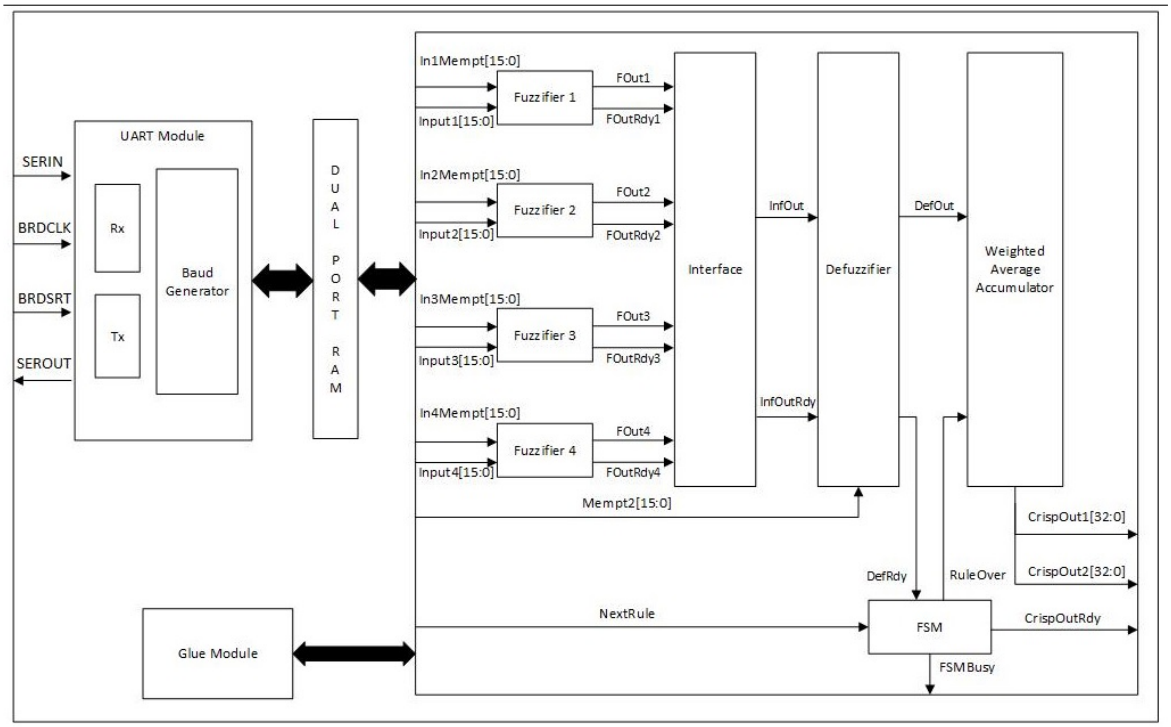


Figure 5.7: Block Diagram of Fuzzy Inference Module

Once new crisp inputs are transferred, the software writes value 0xA3H to RAM location 0x0H, which is the control word as mentioned in Table 5.1. The FLC keeps pooling this value, after reading 0xA3 the FSM initiates the FLC process. The process ends on computing crisp output value software acknowledges this compilation by reading 0xA1 data on RAM location 0x0H. A dual-port RAM is used by ports A and port B interfaced to the UART and the FLC modules respectively. The Xilinx Core Generator tool is used to generate the core of the block RAM module. The functionality of dual ports purges bus sharing issues. The address on port A is set to 0x00 and increments twice for each membership point or crisp input (2 bytes). Arrival of data over UART triggers address, increment and memory write operation.

Following this the rule base is ready for inference processing. The address is set to 0x00H and incremented till it reaches $NoOfBytes$. Here, control registers are to be read first. Then the membership function points are read rule by rule. The location $No_Of_Bytes + 1$ to $No_Of_Bytes + 4$ are used to store crisp output for single output system and $No_Of_Bytes + 1$ to $No_Of_Bytes + 8$ are used to store crisp outputs of 2 outputs system.

$$No\_Of\_Bytes = 2 + 2 * No\_Of\_Inputs + 2 * 3 * No\_Of\_Rules \\ *(No\_Of\_Inputs + No\_Of\_Outputs) \tag{5.6}$$

For defuzzification, a weighted average method is used considering its simple hardware structure. Here FLC is designed to evaluate the output rule wise and accumulate them. It reads the points for the one rule only at a time and computes the output before reading the next set of points. Finally all defuzzified values are accumulated using an accumulator. This architecture reduces the logic utilization of the FLC. The proposed Mealy Finite State Machine in section 3.4.1 is used here to act as control unit and supports partial rules. The rule driven circuit discussed in section 2.5 is reused in 'ReduceRuleAddress Generator' module to reduce rules from the methods MRA-2OMF, MRA-3OMF and MRA-4OMF methods based on users choice.

Mamdani inference method is used here to implement inference processing. The FIP module consists of three sub-modules and control logic. There is a 2-byte register file consisting of 8 registers for storing each rule's output. A 5-byte accumulator is used for multiplication and accumulation. The sub-modules include

I. Fuzzification unit for converting crisp quantity fuzzy variable.

II. An inference engine unit to compute overall control output based on individual contribution of each rule in the reduced rule base.

III. The Defuzzification unit, which converts Fuzzification quantity to the precise quantity.

Figure 5.8 shows the calculation of fuzzified values in hardware by evaluating the crisp input membership degree using membership functions by:
If $Point_1 \leq Input \leq Point_2$

$$\mu = \frac{Input - Point_1}{Point_2 - Point_1} \tag{5.7}$$

If $Point_2 \leq Input \leq Point_3$

$$\mu = \frac{Input - Point_3}{Point_2 - Point_3} \tag{5.8}$$

The max-min composition of the inference model proposed by Mamdani [198] illustrated in Figure 5.9 and Table 5.2 is applied for hardware realization of inference module. The inference module outputs are defined by:

$$\mu O_{m1}^{r1}(y) = min[\mu A1_{j_1}(X1), \mu A2_{k_1}(X2)] \tag{5.9}$$

$$\mu O_{m1}^{r3}(y) = min[\mu A1_{j_3}(X1), \mu A2_{k_3}(X2)] \tag{5.10}$$

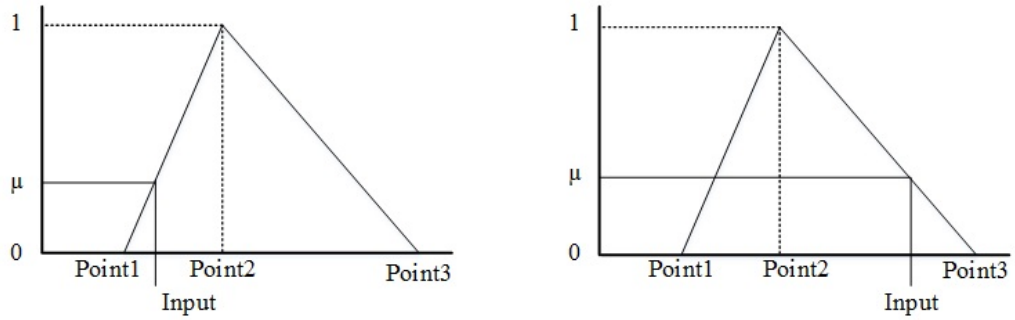


Figure 5.8: Calculation of membership values

Table 5.2: Rule Base of simple FLC

X1	X2	Rule	Y
$A1_{j_1}$	$A2_{k_1}$	r1	O_{m1}
$A1_{j_2}$	$A2_{k_2}$	r2	O_{m2}
$A1_{j_3}$	$A2_{k_3}$	r3	O_{m1}

$$\mu O_{m1}^{r1\&r3}(y) = max\{min[\mu A1_{j_1}(X1), \mu A2_{k_1}(X2)], min[\mu A1_{j_3}(X1), \mu A2_{k_3}(X2)]\} \tag{5.11}$$

$$\mu O_{m2}^{r2}(y) = min[\mu A2_{j_2}(X1), \mu A2_{k_2}(X2)] \tag{5.12}$$

Instead of a composite output membership function weighted average defuzzification is used as shown in Figure 5.10 for its simplicity in hardware implementation as it needs only clipped or scaled output membership functions. This method just takes the peak value of each clipped or scaled output fuzzy sets and builds weighted sum of these peak values given by:

$$Y^* = \frac{\mu O_{m1} * P1 + \mu O_{m2} * P2}{P1 + P2} \tag{5.13}$$

The implemented block diagrams of fuzzifier, inference, and defuzzifier and their top modules in an FPGA are shown in Figure 5.11. Where, three membership points of triangular membership function are used as inputs for fuzzifier and defuzzifier modules with other synchronous signals.

5.6 Validation of Proposed FLC in Practical Systems

At this juncture, it is necessary to implement the proposed algorithm on a programmable hardware and validate the results. To achieve this, the proposed system is used an Intel Corei5-2400 3.1 GHz PC with 4GB memory for optimized rule generation with GA and Xilinx Virtex5 LX110T FPGA for DFLC implementation. DFLC on FPGA is connected to

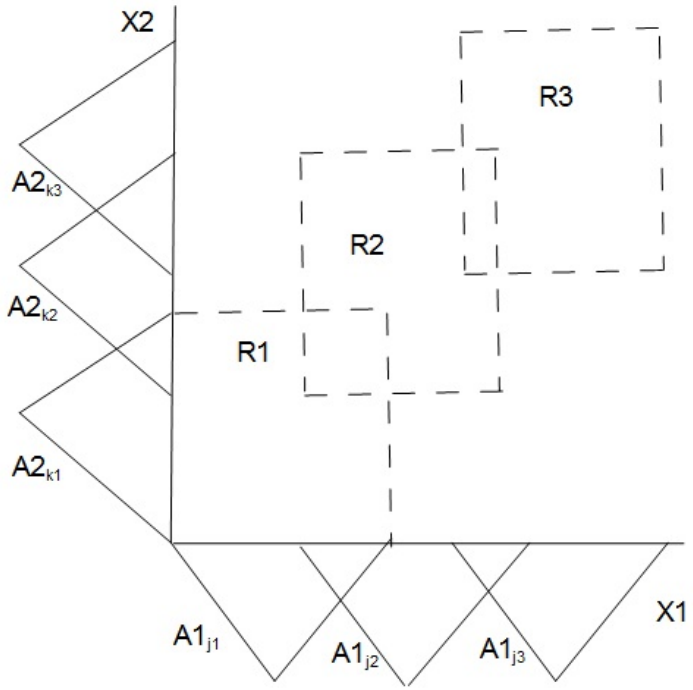


Figure 5.9: Process of fuzzy controller

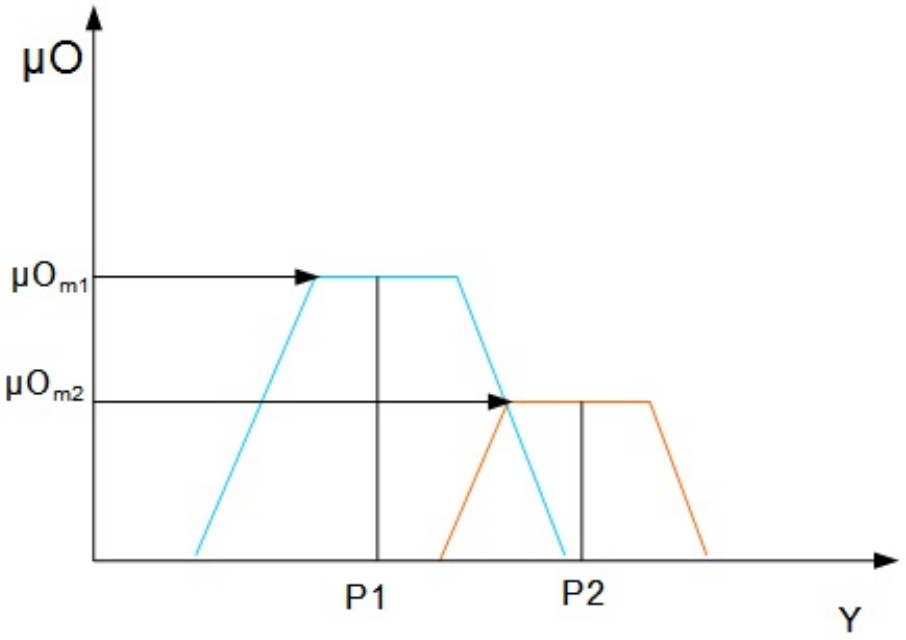


Figure 5.10: Defuzzification

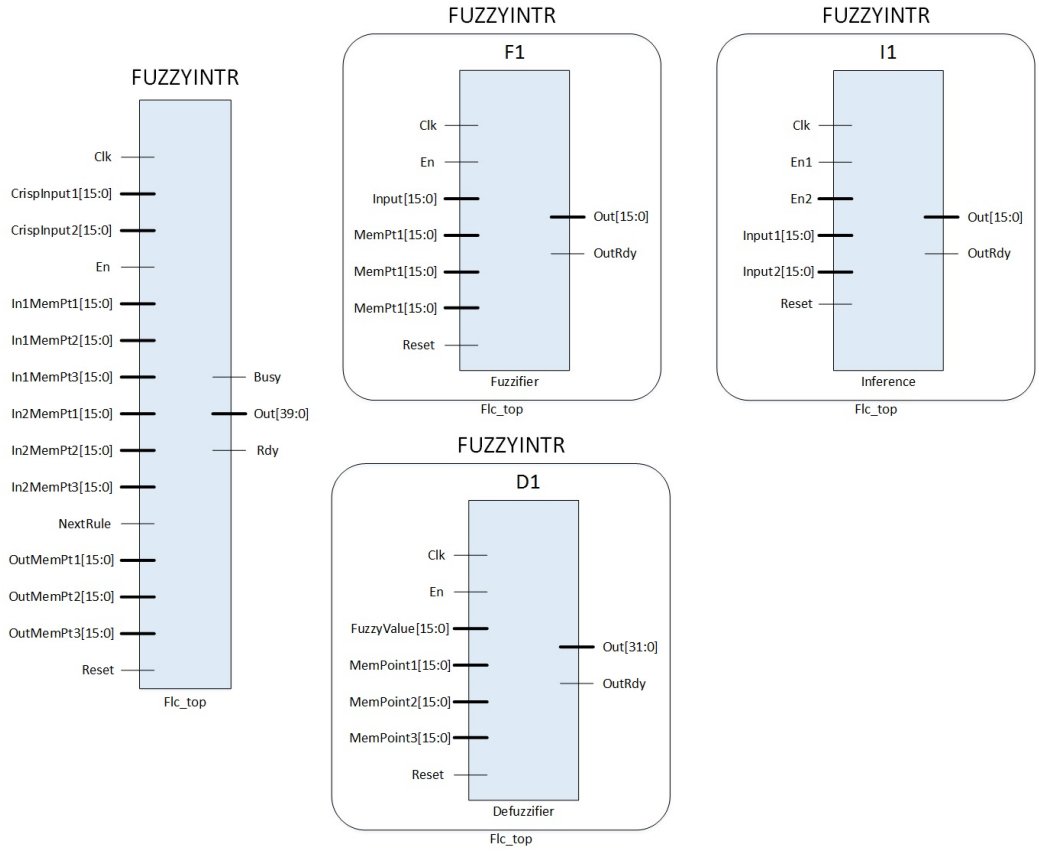


Figure 5.11: Block Diagrams of Fuzzy inference processing top module and fuzzifier, inference and defuzzifier.

Table 5.3: GA-FLC system manual to generate optimized rules for hang data function.

Si No	GA-FLC system parameters	Value
1	Number of points in the data sheet	800
2	Number of features	3
3	Number of designing rules	12
4	Number of populations used for GA optimization	6
5	GA crossover percentage	0.8
6	GA Mutation percentage	0.3
7	Number of GA generations	1000
8	GA Error threshold value	0.001
9	Samples for integration in defuzzification	1000

the PC using on-board UART and provides a platform, which is capable of accepting Fuzzy parameter register file to operate as a standalone tunable FLC.

5.6.1 Hang Data Function [2 input 1 output system]

Hang data function is a test case mathematical function with two inputs and one output. The mathematical representation of the function is given by,

$$y = \{1 + x_1^{-1.5} + x_2^{-2.5}\}^2 \tag{5.14}$$

where

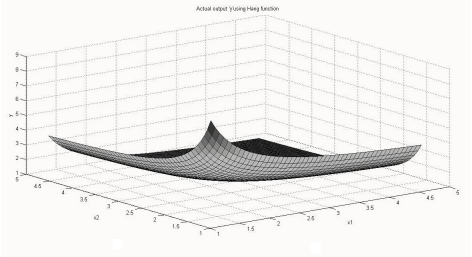
$$1 \leq x_1 \leq 5$$

and

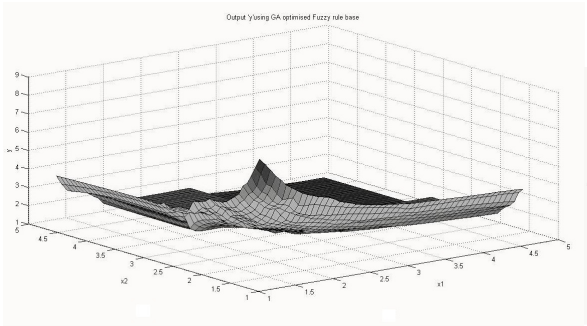
$$1 \leq x_2 \leq 5$$

We obtained 800 input-output data points by sampling the input range $x_1, x_2 \in \{1, 5\}$. Rule base had been generated for this test case and it was optimized using GA. Fuzzy rule base was designed for hang function by generating a training datasheet using the basic relationship at (5.14). The designed dataset is used for training and tuning of the rule base while using GA as the tuning algorithm. A list of GA and fuzzy parameters are provided as shown in Table 5.3, which are considered in the designing of the fuzzy optimized system for Hang data function. Results before and after optimizations of the rule base are as follows: Total mean square error for initial rule base = 283 and Total mean square error for final optimized rule base = 23.

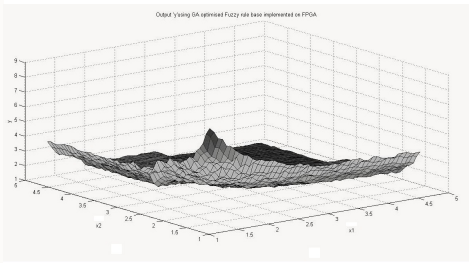
Actual surface plot for hang data function in presented in Figure 5.12a. Figure 5.12b



(a) Hang Function Surface Plot Signal



(b) Hang function approximation with GA trained fuzzy system.



(c) Using GA-FLC on FPGA data

Figure 5.12: Hang function approximation with GA trained fuzzy system on FPGA

shows the surface plot of hang data function using GA based rule base optimized system, where the centroid method is used for defuzzification. The FPGA implemented GA-FLC based hang data function surface plot is shown in Figure 5.12c, where the weighted average method is used to reduce the hardware resource consumption for defuzzification. Figure 5.13 presents the comparison of GA-FLC with 8 rules and artificial Neuro-fuzzy Inference system (ANFIS) with 49 rules that were trained with 10 epochs. From the figure it can be observed that error generated using the GA-FLC provides better results even though 8 rules are used as against 49 rules in ANFIS.

5.6.2 Chaotic Time Series [4 input 1 output system]

Chaotic processes are the type of methods, which have got a disordered mechanism of functioning. These processes have acquired this behavior because of the presence of positive feedback. The analysis of this sort of processes against time results in a form of a random time series can be called as chaotic time series. These systems are never entirely predictable; because of feedback the simulation and the real series will always rapidly diverge.

Let $x(t)$ be a Chaotic Time Series. If t_0 = start time, Δt = time interval, then at time

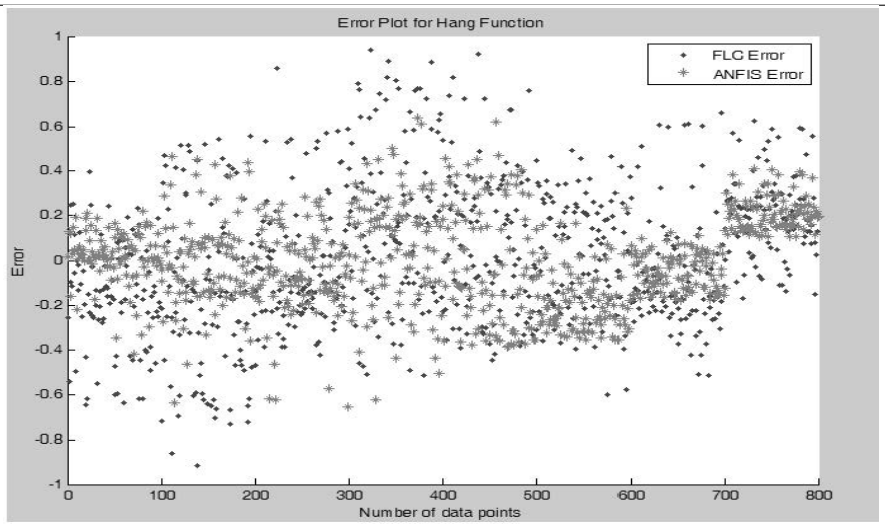


Figure 5.13: GA-FLC versus ANFIS error plot

point $t_k = t_0 + \Delta t * k$, $(0 \leq k \leq n)$. The Chaotic time series data can be represented as,

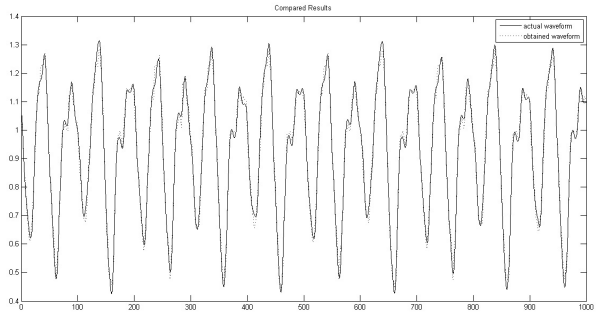
$$X = (x_0, x_1, ...x_n) = (x(t_0), x(t_1), ...x(t_n)) \tag{5.15}$$

Because of the unpredictability of the chaotic time series, it is not possible to mathematically model the series in terms of equations. Hence, the objective is to design a data predictive model T, to predict the value x'_m of the series at time instance t_m based on the available data set, $\{x_k \mid k \leq m\}$ such that $\mid x_m - x'_m \mid$ is as minimum as possible.

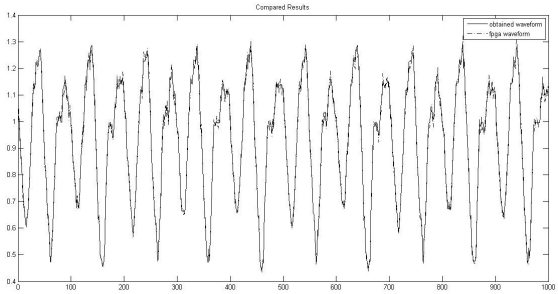
GA optimized fuzzy rule base algorithm is used to design the model for chaotic time series prediction. The chaotic series taken into consideration had Δt=6 and the prediction is done with four previous available data set. The mathematical model is:

$$\begin{aligned} x(t + \Delta t) &= F[x(t), x(t - \Delta t), x(t - 2\Delta t), x(t - 3\Delta t)] \\ \Rightarrow x(t + 6) &= F[x(t), x(t - 6), x(t - 12), x(t - 18)] \end{aligned} \tag{5.16}$$

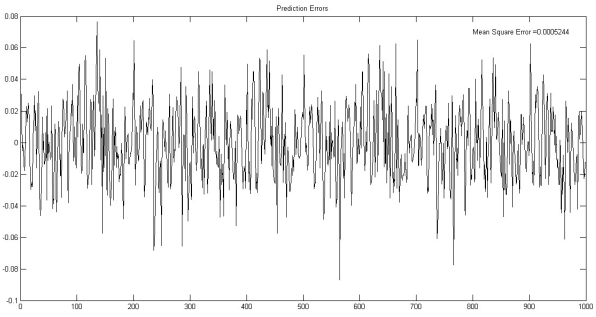
The parameters chosen for designing of the four input one output GA optimized rule base is given in Table 5.3, The rule base tested with its real-time data points and the mean square error obtained for 1000 data points is $5.244 * 10^{-4}$. The optimized fuzzy rule base was sent to the designed FPGA model to predict the data for the same 1000 data points. Comparison of Plots is given in Figure 5.14a, Figure 5.14b, and the error plot is shown in 5.14c.



(a) Desired time series data versus GA rulebase predicted data Signal



(b) GA rulebase predicted data versus GA-FLC on FPGA



(c) Prediction errors between desired time series data and GA predicted data

Figure 5.14: Comparative plots between desired time series data, GA rulebase predicted data, and GA-FLC on FPGA

5.6.3 Plasma Position Control in ADITYA TFTR

Section 3.7 presented the simulink model and system modelling of ADITYA TFTR. The proposed GA based optimized rule base on FPGA is applied on this simulink model. The FLC I/O parameters that is considered are,

i. R_p Error : Radial position Error in Range [-0.05,0.05] with 7 Input MFs (Triangular)

ii. I_p : Plasma Current in Range [5e4,8e4] with 7 Input MFs (Triangular)

iii. u : Control Signal in Range [-60,60] with 7 Output MFs (Triangular)

Figure 5.15 shows the simulation of GA-FLC with other models. The simulation data is used to analyze the control performance. The simulation data is plotted in Figure 5.16 and observed that the proposed GA-FLC shows an improvement in rise time and settling time in accordance with PID controller and proposed DFLC with MRA2-OMF method. Table 5.5 presents the comparative analysis of control parameters. It can be observed that the proposed GA-FLC provides 18% rise time and 31% settling time in comparison to DFLC

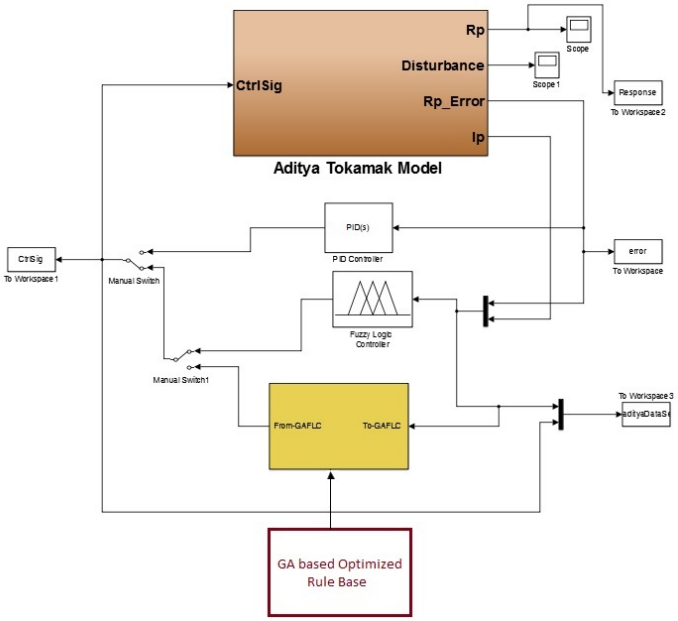


Figure 5.15: Radial Plasma Position Control of Aditya TFTR: HIL Simulation

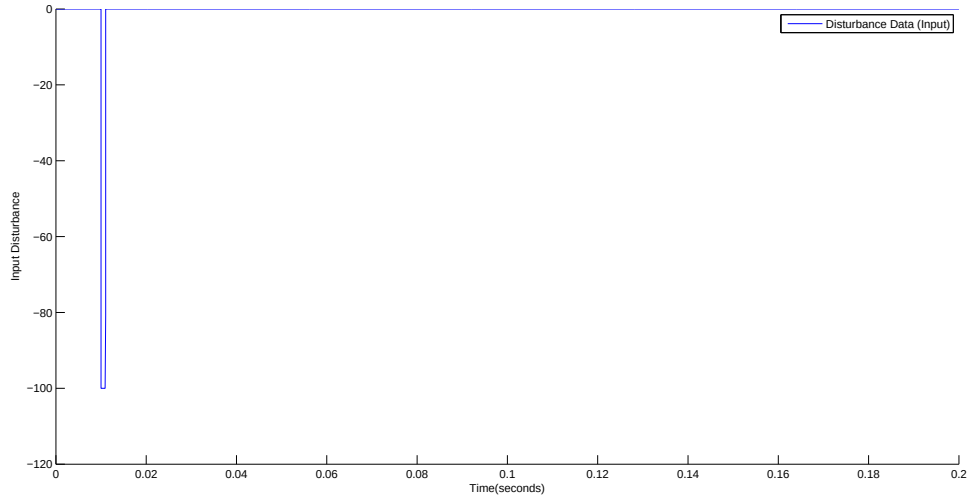
with MRA2-OMF of section 2.5. Table 5.4 provides FPGA performance parameters of GA-FLC with other methods. Where, the GA-FLC outperforms the other proposed designs with its lesser cycle time of 4.06 ns and latency of 60.9 ns with subsequently lesser logic utilization.

Table 5.4: Hardware Implementation: Comparison of proposed methods

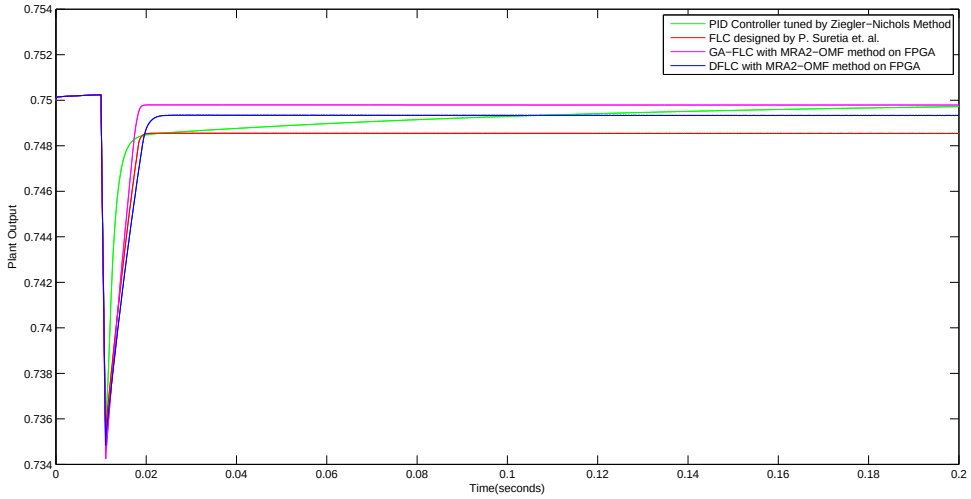
Proposed Methods	DSP48Es	LUTs	Bonded IOBs	Cycle Time	Latency
DFLC with MRA-2OMF	2	4126	6	6.608 ns	99.12 ns
SAIT2FLC with MRA-2OMF	25	8724	10	8.129 ns	267.993 ns
GA-FLC with MRA-2OMF	2	1294	6	4.06 ns	60.9 ns

Table 5.5: Comparison of performance parameters of PID, FLC [1], and DFLC with MRA2-OMF, GA-FLC with MRA2-OMF

Parameters	PID	FLC [1]	DFLC (MRA2-OMF)	GA-FLC (MRA2-OMF)
Rise Time	0.0062	0.0025	0.0023	0.00187
Settling Time	0.1255	NaN	0.0249	0.0170
Overshoot	0	0	0	0
Undershoot	0	0	0	0
Peak	0.7497	0.7483	0.7492	0.7497
Peak Time	0.14	0.02	0.0235	0.0182



(a) Input Disturbance Signal



(b) System Response

Figure 5.16: Performance of various controllers in presence of disturbances in plasma position

5.6.4 Simulation and Hardware Implementation

The implementation of the FLC is carried out by coding each module in Verilog hardware description language integrated with Xilinx foundation ISE 14.2 tool, which supports ISim (Integrated within ISE) is used here for functional verification. The architecture of the FLC is highly pliable as the parameters of the fuzzy logic controller can be changed by modifying registry values and Verilog parameters.

5.6.4.1 Simulation Parameters

i) The UART interface includes receiving and transmits modules that share a single baud generator module. Two constants set the baud rate at FLC top modules, which are calculated as follows:

$$\text{X\_BAUD\_FREQ} = \frac{(16 * \text{BaudRate})}{(\text{GCD}(\text{GlobalClkFreq}, 16 * \text{BaudRate}))} \tag{5.17}$$

$$\text{X\_BAUD\_LIMIT} = \frac{\text{GlobalClkFreq}}{(\text{GCD}(\text{GlobalClkFreq}, 16 * \text{BaudRate}))} - \text{X\_BAUD\_FREQ} \tag{5.18}$$

ii) For 100Mhz of board clock in XUPV5 FPGA board, The calculated parameters set in Verilog is as follows:

`define X_BAUD_FREQ 12'H90

`define X_BAUD_LIMIT 12'H0ba5

iii) The test bench parameterized for total no of inputs and no of membership functions and data bus width for each input and membership value is as below:

`define NO_OF_INPUTS 3'h2

`define NO_OF_MFS 3'h7

`define DATA_BUS_WIDTH 6'h10

`define DEFUZZY_METHOD 2'h0

5.6.4.2 Hardware Implementation

The functional simulations obtained by ISIM 14.2 is presented in Figure 5.17, where the READY signal enable UART transmitter to transmit crisp out data back to MATLAB GUI for display. Figure 5.18 presents, the process of crisp data transfer from FPGA to MATLAB GUI using Chipscope-Pro debugging tool. The implemented platform chosen in this work is the Virtex 5 (LX110T) Xilinx FPGA family included in XUPV5 development board. This FPGA is sufficient enough for implementing all the modules of the FLC addressed in this work. This is possible as the FPGA contains 17,280 slices and 68 DSP48E slices as well as 296 18kb block RAMs etc. Table 5.6 ushered the FPGA utilization to develop the FLC addressed here. The total clocks needed to compete with fuzzy logic inference process (FLIP) with GA reduced 8 rules is 25 clocks with 247 MHz clock frequency, this means that the total

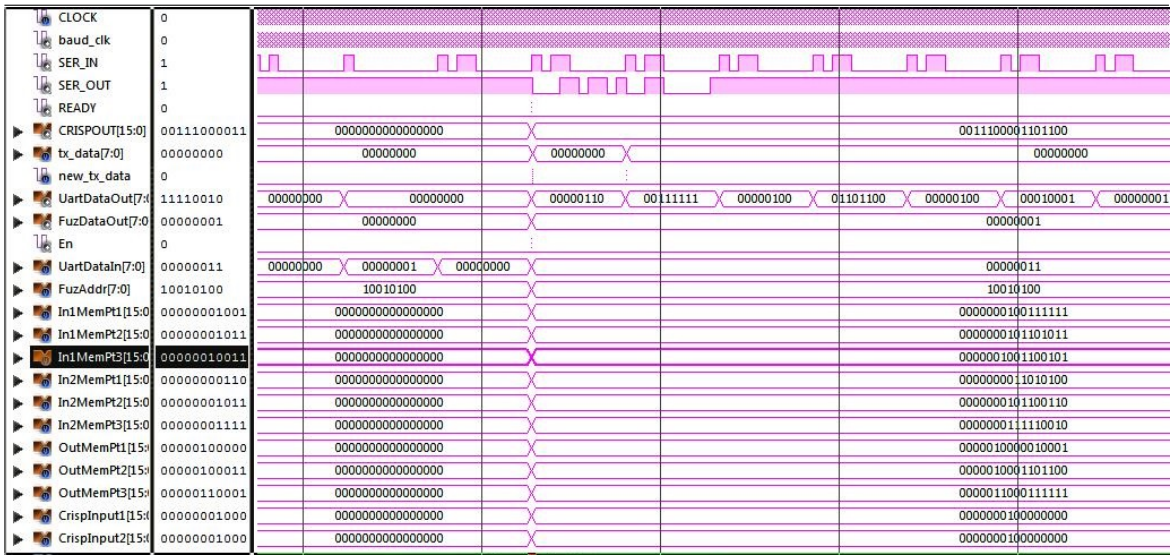


Figure 5.17: The functional simulation waveform obtained by ISim 14.2.

Table 5.6: Device Utilization Summary

Experiment	4 input 1 output system	2 input 1 output system
Selected device	xc5vlx110t-2-ff1136	xc5vlx110t-2-ff1136
Maximum Frequency	213.413Mhz	246.819Mhz
Number of slices	1456 out of 69120 2%	1149 out of 69120 1%
Number of 4 input LUTs	1919 out of 69120 2%	1294 out of 69120 1%
Number of bonded IOBs	6 of 640 1%	6 of 640 1%
Number of MULT18X18s	1 out of 640 1%	1 out of 640 1%
Number of GCLKs	1 out of 32 6%	1 out of 32 6%

time required to complete one FLIP to generate output is $0.10121\mu s$, it is equal to 9880446 (9.8MFLIPS) fuzzy logic inference outputs can be made using the current design.

The GA-FLC using FPGA is proved to be a useful framework, which can be easily used to develop fuzzy logic applications within a short period and the main functional units can be reused without modification and user can only concentrate to provide input-output data sheet, membership values and the number of rules, which increases the design efficiency.

5.7 Summary

A novel approach towards the rule base synthesis for fuzzy systems using genetic algorithm was undertaken. The system is tested with bench mark problems like hang data function and chaotic time series. Compared to an ANFIS model in MATLAB, which used 49 rules and

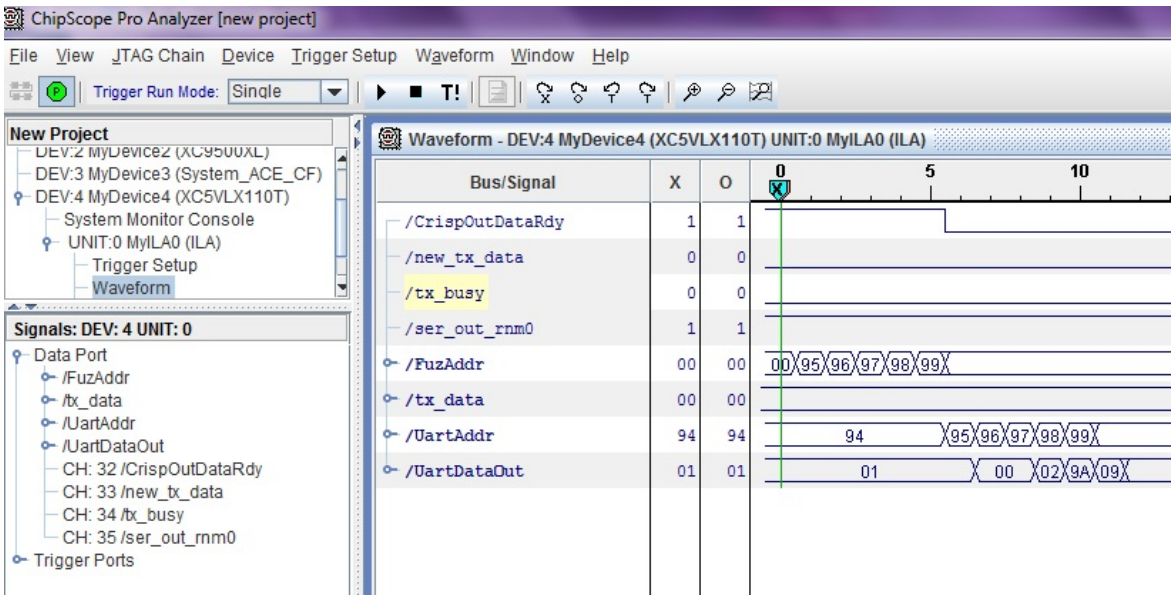


Figure 5.18: Crisp Data output from FPGA using UART captured on the Chipscope - Pro tool

ran for 100 iterations, this novel method provides better results with lesser rules. Further, there is no need for an expert to design the system. An accurate data set of the system/plant under discussion is required only for generating the rule base. The following are the benefits of the algorithm proposed in this work:

i. The user can design required number of rules as per the system memory and time constraints for better response of the system.

ii. The Proposed GA rule base optimization system is easily configured.

iii. Actual process changes can be easily incorporated by redesigning the rule base in short time.

iv. Rule bases can be designed as per the system tolerance limit.

v. As it is mentioned the problems with existing fuzzy systems, this process handles the above issues with automatic rule base designing and auto-optimization process.

vi. The fuzzy rule base is scalable, i.e. any change in the actual process behavior can be taken into consideration easily.

Chapter 6

Conclusion

Preface

This chapter concludes the thesis and summarizes the objectives accomplished to the thesis. It also provides the limitations of this work and future scope of the work through which the current work can be enhanced.

The development of the remotely tunable FLCS in FPGA is discussed in this thesis opens a line of approach to several explorations. This architecture provides a significant number of functionalities to the users along with improved speed to drive a variety of industrial processes. This system performance is compared with MATLAB Fuzzy Logic Toolbox and it is seen that it has the ability to provide good performance. The proposed systems are observed to perform well within multiple testing paradigms mentioned in this work. Thorough investigations have been done using few specific applications to ascertain generality and applicability of DFLCS in control applications, the system designed can be used in a variety of applications.

6.1 Contributions of this thesis

In summary, this research successfully contributes the following

- A DFLC module with MATLAB GUI has been proposed. The FLC can operate as a standalone remotely tunable controller. All existing DFLC do not have user interactions [199, 200] and even though they were developed on field programmable hardware, the system architecture does not permit field programmability of the DFLCS. The proposed MRA2-OMF, MRA3-OMF, and MRA4-OMF based DFLC system can be field programmed through the user interface. The novelty of this development lies in the system architecture which has been elaborated in Chapter 2 and 3.

- A Mealy state machine is proposed and implemented to support special case rules in FLCs. Partial rules have been successfully supported in hardware with reduced logic utilization. A 5% logic utilization saving is observed after merging this state machine with rule reduction techniques. Further, in section 3.4.1, a simple UART based HIL testing process was described that provided performance and timing analysis of the proposed DFLCS.

- The addition of tunability to MRA2-OMF based DFLC slightly reduced by speed but achieved an operating speed of around 21.61 FLIPS. This speed was found to be satisfactory for application related to ADITYA TFTR.

- A GUI based fuzzy validation with test vectors has been analyzed. MicroBlaze processor based IP integration was conducted on simulation and shows that the

proposed DFLC can be easily connected as peripheral to any soft or hard processor with processor logic bus (PLB).

- A Type 2 fuzzifier based on successive approximation method has been proposed and a digital hardware implementation has been carried out. This fuzzier is based on a T2FLC consumed nine clock cycles with 8.129 ns cycle time results in 73 ns latency. Comparative analysis of this technique with other division algorithms type 2 fuzzifiers shows that proposed method has superior speed. An overall operation speed of 3.7 MFLIPS was achieved.

- A novel approach towards the rule base synthesis for fuzzy systems was taken using an evolutionary algorithm based on Genetic Algorithm, for convergence of the rule base to provide superior tuning. Compared to an ANFIS model in MATLAB which used 49 rules and tuned over 100 iterations, it was seen that the proposed method provides superior performance with a lower number of rules. Further, there is no need for an expert to design the system. An accurate data set of the system/plant under discussion is required only for generating the rule base. The major benefits of the algorithm proposed lies in its GA rule base optimization with automatic rule base designing and auto-optimization process.

- The proposed stand-alone tunable fuzzy logic controllers on FPGA have been used to control the radial plasma position of ADITYA TFTR model. The observations obtained from these systems are exciting as they provide better rise time and speedy settling time in comparison to existing control schemes.

6.2 Limitations of this Work

The major limitations of the work reported in this thesis can be summarized as follows:

- DFLC was tested in HIL environment with simulink models. Full-scale hardware was not used in system performance testing. Although the HIL test results are promising, a real-time test will assure system performance.

- In the test procedure, the DFLC was connected to a PC using UART protocol. Other communication protocols like Ethernet, Controller-Area Network (CAN) have

not been integrated and tested. System implementation using these communication interfaces can make the testing complete.

- In this implementation, the security of the data communication network needs to be further investigated to evaluate the network security.

- In successive approximation method, the silicon area was seen to increase with the bit size of type 2 fuzzy process. Hence, to achieve more precision in fuzzy output, the proposed method requires higher silicon area leading to more space and power requirement.

6.3 Future Research Directions

The work presented in this thesis elaborates the design and implementation of FLC-based on MRA-2OMF, MRA-3OMF, and MRA-4OMF. The design discussed DFLCS with special case rule base support. This design has the potential for broad explorations. Some of the significant areas of future work include the following.

- ASIC implementation of DFLCS can investigate the system performance and power consumption with equivalent FPGA using the same process geometry.

- Microblaze driver implementations for light weight IP (LWIP) TCP/IP stack to tune FLC parameters over Ethernet. With Ethernet communication, the proposed DFLC can talk to any other device on the network, and the user can remotely configure the FLC parameters from a workstation.

- The proposed DFLCS architecture is implemented on the type-I and type-II Mamdani fuzzy logic control system. This architecture has been implemented using modular design methodology. The modules in proposed DFLCS can be integrated with a neural network to achieve an FPGA based generic neuro-fuzzy system.

- DFLCS is developed on a programmable hardware. The methodology of DFLC is implemented using an FPGA. A hardware-software codesign of hybrid computing platform with DSP would unleash the complete power of the proposed method. Since DSP provides an efficient implementation of multiplication and accumulation (MAC) and this helps better tunability options for fuzzifier and de-fuzzifier.

- The proposed DFLCs can be tested with other control benchmark problems like automatic cruise control, automotive suspension system, and aircraft pitch system, etc.
- An attempt has been made on rule base generation with Genetic Algorithm for Type-1 FLC in chapter 5. Same rule base can also be applied on Type-2 FLC with a new architecture.
- This concept of DFLCs can be extended to different complex control problems.

References

[1] P. Suratia, J. Patel, R. Rajpal, S. Kotia, and J. Govindarajan, "Fpga based fuzzy logic controller for plasma position control in aditya tokamak," *Fusion Engineering and Design*, vol. 87, no. 11, pp. 1866–1871, 2012.

[2] L. A. Zadeh, "Fuzzy logic, neural networks and soft computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77–84, 1994.

[3] T. J. Ross, *Fuzzy logic with engineering applications*, 3rd ed. John Wiley & Sons, Ltd, 2010.

[4] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[5] H. Nguyen, N. Prasad, C. Walker, and E. Walker, *A First Course in Fuzzy and Neural Control*. Chapman and Hall, CRC press, 2003.

[6] S. Bogdan and Z. Kovacic, *Fuzzy Controller Design: Theory and Applications*, 1st ed. CRC press, Taylor and Francis, 2006.

[7] K. Passino and S. Yurkovich, *Fuzzy Control*. Addisson Wisely, 1998.

[8] Z. Kovacic and S. Bogdan, *Fuzzy controller design: theory and applications*. CRC press, 2005, vol. 19.

[9] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[10] P. Londhe, B. Patre, and A. Tiwari, "Design of single-input fuzzy logic controller for spatial control of advanced heavy water reactor," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 901 – 911, 2014.

[11] A. El Khateb, N. Abd Rahim, J. Selvaraj, and M. Uddin, "Fuzzy-logic-controller-based sepic converter for maximum power point tracking," *IEEE Transactions on Industry Applications*, vol. 50, no. 4, pp. 2349 – 2358, 2014.

[12] S. Seyedtabaii and A. Khalaji, "Single chip digital implementation CMOS of a reconfigurable fuzzy logic traffic controller," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 27, no. 2, pp. 921 – 928, 2014.

[13] H. Peyravi, A. Khoei, and K. Hadidi, "Design of an analog CMOS fuzzy logic controller chip," *Fuzzy Sets and Systems*, vol. 132, no. 2, pp. 245 – 260, 2002.

[14] T. Korol, "A fuzzy logic model for forecasting exchange rates," *Knowledge-Based Systems*, vol. 67, pp. 49–60, 2014.

[15] W. L. Tung, C. Quek, and P. Cheng, "GenSo-EWS: a novel neural-fuzzy based early warning system for predicting bank failures." *Neural networks : the official journal of the International Neural Network Society*, vol. 17, no. 4, pp. 567–587, 2004.

[16] Y. Yoshida, "The valuation of European options in uncertain environment," *European Journal of Operational Research*, vol. 145, no. 1, pp. 221–229, 2003.

[17] P. Dostal, *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2013.

[18] O. Linda and M. Manic, "Uncertainty-robust design of interval type-2 fuzzy logic controller for delta parallel robot," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 661–670, 2011.

[19] J. J. Acevedo, B. Arrue, J. M. Diaz-Banez, I. Ventura, I. Maza, and A. Ollero, "One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74, no. 1-2, pp. 269–285, 2014.

[20] T. Das and I. Kar, "Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 501–510, 2006.

[21] K. Lochan and B. Roy, "Control of two-link 2-dof robot manipulator using fuzzy logic techniques: A review," in *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*. Springer, 2015, pp. 499–511.

[22] S. Gopinath, I. Kar, and R. Bhatt, "Experience inclusion in iterative learning controllers: Fuzzy model based approaches," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 4, pp. 578–590, 2008.

[23] C.-H. Wang and C.-C. Wang, "Finding the real surge boundaries of turbo-charged automobiles using intelligent fuzzy reasoning technique," *International Journal of Fuzzy Systems*, vol. 17, no. 2, pp. 224–235, 2015.

[24] M. D. Baldania, D. A. Sawant, and A. B. Patki, "Fuel saving of an automobile using fuzzy logic based embedded controller," in *IEEE International Conference on Advanced Communications, Control and Computing Technologies*. IEEE, 2014, pp. 136–140.

[25] N. C. Basjaruddin, Kuspriyanto, D. Saefudin, E. Rakhman, and A. M. Ramadlan, "Overtaking assistant system based on fuzzy logic," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 13, no. 1, pp. 76–84, 2015.

[26] H. Li, X. Jing, H.-K. Lam, and P. Shi, "Fuzzy sampled-data control for uncertain vehicle suspension systems." *IEEE transactions on cybernetics*, vol. 44, no. 7, pp. 1111–1126, 2014.

[27] S. Bogdan, B. Birgmajer, and Z. Kovačić, "Model predictive and fuzzy control of a road tunnel ventilation system," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 5, pp. 574–592, 2008.

[28] C. Bhende, S. Mishra, and S. Jain, "TS-fuzzy controlled active power filter for load compensation," *IEEE Transactions on Power Delivery*, vol. 21, no. 3, pp. 1459–1465, 2006.

[29] C. W. Lou and M. C. Dong, "A novel random fuzzy neural networks for tackling uncertainties of electric load forecasting," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 34–44, 2015.

[30] A. Al Nabulsi and R. Dhaouadi, "Efficiency optimization of a DSP-based standalone PV system using fuzzy logic and dual-MPPT control," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 573–584, 2012.

[31] H. Pan, H. Wong, V. Kapila, and M. S. de Queiroz, "Experimental validation of a nonlinear backstepping liquid level controller for a state coupled two tank system," *Control Engineering Practice*, vol. 13, no. 1, pp. 27–40, 2005.

[32] F. Aqlan and E. Mustafa Ali, "Integrating lean principles and fuzzy bow-tie analysis for risk assessment in chemical industry," *Journal of Loss Prevention in the Process Industries*, vol. 29, no. 1, pp. 39–48, 2014.

[33] N. Lerkkasemsan and L. E. Achenie, "Pyrolysis of biomass fuzzy modeling," *Renewable Energy*, vol. 66, pp. 747–758, 2014.

[34] A. Shamiri, S. W. Wong, M. F. Zanil, M. A. Hussain, and N. Mostoufi, "Modified two-phase model with hybrid control for gas phase propylene copolymerization in fluidized bed reactors," *Chemical Engineering Journal*, vol. 264, pp. 706–719, 2015.

[35] F.-C. Liu, L.-H. Liang, and J.-J. Gao, "Fuzzy PID control of space manipulator for both ground alignment and space applications," *International Journal of Automation and Computing*, vol. 11, no. 4, pp. 353 – 360, 2014.

[36] Y. Zhang, T. Yang, C. Li, S. Liu, C. Du, M. Li, and H. Sun, "Fuzzy-PID control for the position loop of aerial inertially stabilized platform," *Aerospace Science and Technology*, vol. 36, pp. 21 – 26, 2014.

[37] H. Gao, J. Liu, Y. Li, K. Hong, and Y. Zhang, "Dual-layer fuzzy control architecture for the CAS rover arm," *International Journal of Control, Automation, and Systems*, vol. 13, no. 5, pp. 1262 – 1271, 2015.

[38] S. Yauldegar, H. Ghiasi, M. H. Mazloom, A. Sahamijoo, M. R. Avazpour, and F. Piltan, "Trajectory tracking control of multi degrees of freedom joints: Robust fuzzy logic-based sliding mode approach," *International Journal of Control and Automation*, vol. 7, no. 12, pp. 323 – 338, 2014.

[39] T. Sreenuch, F. Khan, and J. Li, "Particle filter with operational-scalable Takagi-Sugeno fuzzy degradation model for filter-clogging prognosis," *Journal of Aerospace Information Systems*, vol. 12, no. 5, pp. 398 – 412, 2015.

[40] M. E. Ooi, M. Sayuti, and A. A. Sarhan, "Fuzzy logic-based approach to investigate the novel uses of nano suspended lubrication in precise machining of aerospace tempered grade 6061," *Journal of Cleaner Production*, vol. 89, pp. 286 – 295, 2015.

[41] I. A. Zammar, I. Mantegh, M. S. Huq, A. Yousefpour, and M. Ahmadi, "Intelligent thermal control of resistance welding of fiberglass laminates for automated manufacturing," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1069–1078, 2015.

[42] S. Rajak and S. Vinodh, "Application of fuzzy logic for social sustainability performance evaluation: a case study of an Indian automotive component manufacturing organization," *Journal of Cleaner Production*, pp. 108–121, 2015.

[43] J. Gokulachandran and K. Mohandas, "Prediction of cutting tool life based on Taguchi approach with fuzzy logic and support vector regression techniques," *International Journal of Quality & Reliability Management*, vol. 32, no. 3, pp. 270–290, 2015.

[44] Z. Zhou, F. Chu, A. Che, and M. Zhou, "Constraint and fuzzy logic-based optimization of hazardous material transportation via lane reservation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 847 – 857, 2013.

[45] K. Noori and K. Jenab, "Fuzzy reliability-based traction control model for intelligent transportation systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 1, pp. 229 – 234, 2013.

[46] S. K. Singh and S. P. Yadav, "Efficient approach for solving type-1 intuitionistic fuzzy transportation problem," *International Journal of Systems Assurance Engineering and Management*, vol. 6, no. 3, pp. 259 – 267, 2015.

[47] P. Kundu, S. Kar, and M. Maiti, "Fixed charge transportation problem with type-2 fuzzy variables," *Information Sciences*, vol. 255, pp. 170 – 186, 2014.

[48] S. Bogdan, B. Birgmajer, and Z. Kovačić, "Model predictive and fuzzy control of a road tunnel ventilation system," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 5, pp. 574–592, 2008.

[49] K. Kumar, S. Deep, S. Suthar, M. Dastidar, and T. Sreekrishnan, "Application of fuzzy inference system (FIS) coupled with Mamdani's method in modelling and optimization of process parameters for biotreatment of real textile wastewater," *Desalination and Water Treatment*, pp. 1–8, 2015.

[50] A. Kumar, A. Khosla, J. S. Saini, and S. S. Sidhu, "Range-free 3D node localization in anisotropic wireless sensor networks," *Applied Soft Computing*, vol. 34, pp. 438–448, 2015.

[51] X. Sun, S. Chung, and F. T. Chan, "Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits," *Transportation Research Part E: Logistics and Transportation Review*, vol. 79, pp. 110–127, 2015.

[52] H. Boumaaraf, A. Talha, and O. Bouhali, "A three-phase NPC grid-connected inverter for photovoltaic applications using neural network MPPT," *Renewable and Sustainable Energy Reviews*, vol. 49, pp. 1171–1179, 2015.

[53] V. K. Jadoun, N. Gupta, K. Niazi, and A. Swarnkar, "Modulated particle swarm optimization for economic emission dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 80–88, 2015.

[54] P. Dutta, O. Mishra, and M. Naskar, "A reviw of operational earthquake forecasting methodologies using linguistic fuzzy rule-based models from imprecise data with weighted regression approach," *Journal of Sustainability Science and Management*, vol. 8, no. 2, pp. 220–235, 2013.

[55] H. Lam and F. F. Leung, "Fuzzy controller with stability and performance rules for nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, no. 2, pp. 147–163, 2007.

[56] M. Togai and H. Watanabe, "A VLSI implementation of a fuzzy-inference engine: toward an expert system on a chip," *Information Sciences*, vol. 38, no. 2, pp. 147 – 163, 1986.

[57] T. Yamakawa, "A fuzzy inference engine in nonlinear analog mode and its application to a fuzzy logic control," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 496 – 522, 1993.

[58] I. Baturone, S. Sanchez-Solano, A. Barriga, and J. L. Huertas, "CMOS fuzzy controllers implemented as mixed-signal ICs," in *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3. IEEE, 1996, pp. 422–425.

[59] L. Lemaitre, M. Patyra, and D. Mlynek, "Analysis and design of CMOS fuzzy logic controller in current mode," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 317 – 322, 1994.

[60] T. Yamakawa, "A fuzzy inference engine in nonlinear analog mode and its application to a fuzzy logic control," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 496 – 522, 1993.

[61] L. Peters, S. Guo, and R. Camposano, "A novel analog fuzzy controller for intelligent sensors," *Fuzzy Sets and Systems*, vol. 70, no. 2-3, pp. 235 – 247, 1995.

[62] G. Marshall and S. Collins, "Fuzzy logic architecture using subthreshold analogue floating-gate devices," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 1, pp. 32 – 43, 1997.

[63] I. Baturone, S. Sanchez-Solano, A. Barriga, and J. Huertas, "Implementation of CMOS fuzzy controllers as mixed-signal integrated circuits," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 1, pp. 1 – 19, 1997.

[64] O. Landolt, "Low-power analog fuzzy rule implementation based on a linear mos transistor network," in *Microelectronics for Neural Networks, 1996., Proceedings of Fifth International Conference on*. IEEE, 1996, pp. 86–93.

[65] K. Tsukano and T. Inoue, "Synthesis of operational transconductance amplifier-based analog fuzzy functional blocks and its application," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 61 – 68, 1995.

[66] T. Inoue, T. Motomura, R. Matsuo, and F. Ueno, "New OTA-based analog circuits for fuzzy membership functions and max/min operations," *IEICE Transactions*, vol. E74, no. 11, pp. 3619 – 3621, 1991.

[67] U. Cilingiroglu, B. Pamir, Z. Gunay, and F. Dulger, "Sampled-analog implementation of application-specific fuzzy controllers," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 431 – 442, 1997.

[68] J. Huertas, S. Sanchez-Solano, A. Barriga, and I. Baturone, "A fuzzy controller using switched-capacitor techniques," in *Fuzzy Systems, 1993., Second IEEE International Conference on*. IEEE, 1993, pp. 516–520.

[69] Y.-P. Ko, Y.-S. Lee, and W.-H. Chao, "Analysis, design and implementation of fuzzy logic controlled quasi-resonant zero-current switching switched-capacitor bidirectional converter," *IET Power Electronics*, vol. 4, no. 6, pp. 683 – 692, 2011.

[70] E. Pierzchala, M. Perkowski, and S. Grygiel, "A field programmable analog array for continuous, fuzzy, and multi-valued logic applications," in *IEEE International Symposium on Multiple-Valued Logic(ISMVL2004)*, 1994.

[71] J. F. M. Amaral, J. L. M. Amaral, C. C. Santini, M. A. Pacheco, R. Tanscheit, and M. H. Szwarcman, "Intrinsic evolution of analog circuits on a programmable analog multiplexer array," in *International Conference on Computational Science*. Springer, 2004, pp. 1273–1280.

[72] S. Ionita and E. Sofron, "Field-programmable analog filters array with applications for fuzzy inference systems," in *International Conference on Hybrid Intelligent Systems*, 2005, pp. 470–471.

[73] T. Miki and T. Yamakawa, "Fuzzy inference on an analog fuzzy chip," *IEEE Micro*, vol. 15, no. 4, pp. 8 – 18, 1995.

[74] S. Guo, L. Peters, and H. Surmann, "Design and application of an analog fuzzy logic controller," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 4, pp. 429 – 438, 1996.

[75] J. Fattaruso, S. Mahant-Shetti, and J. Brock Barton, "A fuzzy logic inference processor," New York, NY, USA, 1993, pp. 210 – 214.

[76] M. M. Eichfeld, H. Lohner, "Architecture of a CMOS fuzzy logic controller with optimized organisation and operator design," in *International Conference on Fuzzy Systems*, 1992, pp. 1317–1323.

[77] H. Watanabe, J. Symon, W. Dettloff, and K. Yount, "VLSI fuzzy chip and inference accelerator board systems," in *International Symposium on Multiple- Valued Logic.IEEE*, 1991, pp. 120 – 127.

[78] S.-H. Huang and J.-Y. Lap, "A high-speed VLSI fuzzy inference processor for trapezoid-shaped membership functions," *Journal of Information Science and Engineering*, vol. 21, no. 3, pp. 607 – 626, 2005.

[79] D. Falchieri, A. Gabrielli, and E. Gandolfi, "Very fast rate 2-input fuzzy processor for high energy physics," *Fuzzy Sets and Systems*, vol. 132, no. 2, pp. 261 – 272, 2002.

[80] M. Haji Seyed Javadi, H. R. Mahdiani, and E. Zeinali Kh, "A hardware oriented fuzzification algorithm and its VLSI implementation," *Soft Computing*, vol. 17, no. 4, pp. 683 – 690, 2013.

[81] W. Dettloff and K. Yount, "A VLSI fuzzy logic inference engine for real-time process control," in *Custom Integrated Circuits Conference, Proceedings of the IEEE*, 1989, pp. 12–14.

[82] M. Patyra, J. Grantner, and K. Koster, "Digital fuzzy logic controller: design and implementation," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 4, pp. 439 – 459, 1996.

[83] M. Jacomet and R. Walti, "A VLSI fuzzy processor with parallel rule execution," in *Proc. 5th Int. Conf. on Fuzzy Systems*. Citeseer, 1996, pp. 554–558.

[84] H. Eichfeld, M. Klimke, M. Menke, J. Nolles, and T. Kunemund, "A general-purpose fuzzy inference processor," in *IEEE Micro*, 1994, pp. 310 – 317.

[85] N. Yubazaki, M. Otani, A. Muto, T. Ashida, J. Yi, K. Hirota, and Y. Shi, "Fuzzy inference chip FZP-0401A based on interpolation algorithm," *Fuzzy Sets and Systems*, vol. 98, no. 3, pp. 299–310, 1998.

[86] G. Cardarilli, M. Re, and R. Lojacono, "VLSI implementation of a real time fuzzy processor," *Journal of Intelligent and Fuzzy Systems*, vol. 6, no. 3, pp. 389–401, 1998.

[87] N. Evmorfopoulos and J. Avaritsiotis, "An adaptive digital fuzzy architecture for application-specific integrated circuits," *Active and Passive Electronic Components*, vol. 25, no. 4, pp. 289 – 306, 2002.

[88] H. Sun and J. Wu, "Plating pulse switching power based on a CPLD," *Electrochimica Acta*, vol. 105, pp. 342–346, 2013.

[89] J. Xue, L. Sun, C. Qiao, and H. Qian, "Research on high-speed fuzzy reasoning with CPLD for fault diagnosis expert system," in *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE, 2009, pp. 564–568.

[90] B. Adhavan and C. S. Ravichandran, "FPGA implementation to minimize torque ripples in permanent magnet synchronous motor driven by field oriented control using fuzzy logic controller," *Journal of Theoretical and Applied Information Technology*, vol. 61, no. 2, pp. 369–377, 2014.

[91] A. Benzekri and A. Azrar, "FPGA Based design process of a fuzzy logic controller for a Dual-Axis sun tracking system," *Arabian Journal for Science and Engineering*, vol. 39, no. 8, pp. 6109–6123, 2014.

[92] M. P. S. Dos Santos and J. a. F. Ferreira, "Novel intelligent real-time position tracking system using FPGA and fuzzy logic," *ISA Transactions*, vol. 53, no. 2, pp. 402–414, 2014.

[93] A. Messai, A. Mellit, A. M. Pavan, A. Guessoum, and H. Mekki, "FPGA-based implementation of a fuzzy controller MPPT for photovoltaic module ," *Energy Conversion and Management*, vol. 52, no. 7, pp. 2695 – 2704, 2011.

[94] M. D. Schrieber and M. Biglarbegian, "Hardware implementation and performance comparison of interval type-2 fuzzy logic controllers for real-time applications," *Applied Soft Computing*, vol. 32, pp. 175–188, 2015.

[95] H. Tamukoh, K. Horio, and T. Yamakawa, "A bit-shifting-based fuzzy inference for self-organizing relationship (SOR) network," *IEICE Electronics Express*, vol. 4, no. 2, pp. 60–65, 2007.

[96] D. Hung and W. Zajak, "Design and implementation of a hardware fuzzy inference system," *Information Sciences, Applications*, vol. 3, no. 3, pp. 193 – 207, 1995.

[97] T. Hollstein, S. Halgamuge, and M. Glesner, "Computer-aided design of fuzzy systems based on generic vhdl specifications," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 4, pp. 403 – 417, 1996.

[98] R. Amore, O. Saotome, and K. Kienitz, "A two-input, one-output bit-scalable architecture for fuzzy processors," *IEEE Design and Test of Computers*, vol. 18, no. 4, pp. 56 – 64, 2001.

[99] A. Di Stefano and C. Giaconia, "An FPGA-based adaptive fuzzy coprocessor," in *International Work-Conference on Artificial Neural Networks*. Springer, 2005, pp. 590–597.

[100] A. Barriga, S. Sanchez-Solano, P. Brox, A. Cabrera, and I. Baturone, "Modelling and implementation of fuzzy systems based on vhdl," *International Journal of Approximate Reasoning*, vol. 41, no. 2, pp. 164 – 178, 2006.

[101] R. Castaneda-Miranda, E. V.-R. Jr., R. del Rocio Peniche-Vera, and G. Herrera-Ruiz, "Fuzzy greenhouse climate control system based on a field programmable gate array," *Biosystems Engineering*, vol. 94, no. 2, pp. 165 – 177, 2006.

[102] S. Dick, V. Gaudet, and H. Bai, "Bit-serial arithmetic: A novel approach to fuzzy hardware implementation," in *Fuzzy Information Processing Society, 2008. NAFIPS 2008. Annual Meeting of the North American*. IEEE, 2008, pp. 1–6.

[103] S. Tzafestas, K. Deliparaschos, and G. Moustris, "Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of system on a chip," *Robotics and Autonomous Systems*, vol. 58, no. 8, pp. 1017 – 1027, 2010.

[104] J. Binfet and B. Wilamowski, "Microprocessor implementation of fuzzy systems and neural networks," in *International Joint Conference on Neural Networks (IJCNN 01)*, vol. 1, 2001, pp. 234 – 239.

[105] G. Nhivekar, S. Nirmale, and R. Mudholker, "Implementation of fuzzy logic control algorithm in embedded microcomputers for dedicated application," *International Journal of Engineering, Science and Technology*, vol. 3, pp. 276 – 283, 2012.

[106] N. Eskandarian, Y. A. Beromi, and S. Farhangi, "Improvement of dynamic behavior of shunt active power filter using fuzzy instantaneous power theory," *Journal of Power Electronics*, vol. 14, no. 6, pp. 1303–1313, 2014.

[107] S. Rafa, A. Larabi, L. Barazane, M. Manceur, N. Essounbouli, and A. Hamzaoui, "Implementation of a new fuzzy vector control of induction motor." *ISA transactions*, vol. 53, no. 3, pp. 744–754, 2014.

[108] P. Maji, *On Design and Implementation of Generic Fuzzy Logic Controllers*. NIT Rourkela, 2015.

[109] H. Kahveci, H. Okumus, and M. Ekici, "Improved brushless DC motor speed controller with digital signal processor," *Electronics Letters*, vol. 50, no. 12, pp. 864–866, 2014.

[110] S. Gai, P. Liu, J. Liu, and X. Tang, "A method for banknote feature extraction based on Haar wavelet and fuzzy logic," *Gaojishu Tongxin/Chinese High Technology Letters*, vol. 20, no. 11, pp. 1149–1155, 2010.

[111] N. Lall, Xilinx, "FPGAs and DSPs - What Makes Sense for Your Design?" *RTC Magazine*, p. 1, 2005.

[112] M. Parker, "FPGA versus DSP design reliability and maintenance," in *Design*, no. May, 2010, pp. 1–4.

[113] R. Schneiderman, "Automotive industry is a key component to the success of the DSP sector [Special Reports]," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 18–21, 2014.

[114] L. Zhao, F. Hicks, and A. Robinson Fayek, "Long lead forecasting of spring peak runoff using mamdani-type fuzzy logic systems at hay river NWT," *Canadian Journal of Civil Engineering*, vol. 42, no. 9, pp. 665 – 674, 2015.

[115] M. Muslim, I. Kurniawati, and E. Sugiharti, "Expert system diagnosis chronic kidney disease based on mamdani fuzzy inference system," *Journal of Theoretical and Applied Information Technology*, vol. 78, no. 1, pp. 70 – 75, 2015.

[116] F. Camastra, A. Ciaramella, V. Giovannelli, M. Lener, V. Rastelli, A. Staiano, G. Staiano, and A. Starace, "A fuzzy decision system for genetically modified plant environmental risk assessment using mamdani inference," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1710 – 1716, 2015.

[117] I. Kalaykov and G. Tolt, "Fast fuzzy signal and image processing hardware," in *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*. IEEE, 2002, pp. 7–12.

[118] Xilinx, "Chipscope pro software and cores," in *User Guide*, 2012, pp. 33–129.

[119] D. Kim, "An implementation of fuzzy logic controller on the reconfigurable FPGA system," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 3, pp. 703 – 715, 2000.

[120] F. Taeed, Z. Salam, and S. Ayob, "FPGA implementation of a single-input fuzzy logic controller for boost converter with the absence of an external analog-to-digital converter," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1208 – 1217, 2012.

[121] Y. Sun, S. Tang, Z. Meng, Y. Zhao, and Y. Yang, "A scalable accuracy fuzzy logic controller on FPGA," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6658 – 6673, 2015.

[122] Mathworks Inc., "Build Mamdani Systems," 2010. [Online]. Available: http://in.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html

[123] R. S. Smith and J. Doyle, "The Two Tank Experiment: A Benchmark Control Problem," in *American Control Conference*, Atlanta, Ga, USA, 1988, pp. 2026–2031.

[124] H. Li, N. Godfrey, and Y. Ji, "A fuzzy logic beam-and-ball controller prototype," *IEEE Micro*, vol. 15, no. 6, pp. 64 –67, 1995.

[125] S. B. Bhatt, D. Bora, and B. N. Buch, "ADITYA: The first Indian tokamak," *Indian Journal of Pure and Applied Physics*, vol. 7, no. 9, pp. 710–742, 1989.

[126] C. M. Bishop, P. S. Haynes, M. E. U. Smith, T. N. Todd, D. L. Trotman, and C. G. Windsor, "Real-time control of a Tokamak plasma using neural networks," in *Neural Computation*, 1995, vol. 7, no. 1, pp. 1007–1013.

[127] J. Morelli, A. Hirose, and H. Wood, "Fuzzy-logic-based plasma-position controller for STOR-M," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 2, pp. 328–337, 2005.

[128] I. Bandyopadhyay, S. P. Deshpande, and S. Chaturvedi, "Design analysis of plasma position control in SST1," *Fusion Engineering and Design*, vol. 54, no. 2, pp. 151–166, 2001.

[129] I. Bandyopadhyay, D. Raju, and S. Deshpande, "Modelling of advanced plasma configurations in SST-1 tokamak," *Nuclear Fusion*, vol. 46, no. 3, pp. 62–71, 2006.

[130] I. Bandyopadhyay, S. M. Ahmed, P. K. Atrey, S. B. Bhatt, R. Bhattacharya, M. B. Chaudhury, S. P. Deshpande, C. N. Gupta, R. Jha, Y. S. Joisa, V. Kumar, R. Manchanda, D. Raju, C. V. S. Rao, and P. Vasu, "Modelling of Ohmic discharges in ADITYA tokamak using the Tokamak Simulation Code," *Plasma Physics and Controlled Fusion*, vol. 46, no. 9, pp. 1443–1453, 2004.

[131] A. Sutradhar, A. Chaudhuri, S. Bera, and S. Sadhu, "Analysis and design of an optimal pid controller for insulin dispenser system," *Journal of the Institution of Engineers(India): Electrical Engineering Division*, vol. 82, pp. 304–313, 2002.

[132] V. S. Mukhovatov and V. D. Shafranov, "Plasma equilibrium in a Tokamak," *Nuclear Fusion*, vol. 11, no. 6, p. 605, 1971.

[133] N. N. Karnik and J. M. Mendel, "Introduction to type-2 fuzzy logic systems," in *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 915–920.

[134] H. Hagras, "Type-2 FLCs: A new generation of fuzzy controllers," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30 – 43, 2007.

[135] H.-K. Lam and L. D. Seneviratne, "Stability analysis of interval type-2 fuzzy-model-based control systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 3, pp. 617–628, 2008.

[136] J. Mendel and R. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 117 – 127, 2002.

[137] N. Karnik and J. Mendel, "Operations on type-2 fuzzy sets," *Fuzzy Sets and Systems*, vol. 122, no. 2, pp. 327 – 348, 2001.

[138] D. Wu, "On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 5, pp. 832 – 848, 2012.

[139] S. Raju and G. Pillai, "Design and real time implementation of type-2 fuzzy vector control for DFIG based wind generators," *Renewable Energy*, vol. 88, pp. 40 – 50, 2016.

[140] A. El-Nagar and M. El-Bardini, "Hardware-in-the-loop simulation of interval type-2 fuzzy PD controller for uncertain nonlinear system using low cost microcontroller," *Applied Mathematical Modelling*, vol. 40, no. 3, pp. 2346 – 2355, 2016.

[141] L. Leottau and M. A. Melgarejo, "Implementing an interval type-2 fuzzy processor onto a DSC 56F8013." in *FUZZ-IEEE*, 2010, pp. 1–4.

[142] M. Melgarejo and C. A. Pena-Reyes, "Implementing interval type-2 fuzzy processors," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 63 – 71, 2007.

[143] U. Pareek and I. N. Kar, "Estimating compressor discharge pressure of gas turbine power plant using type-2 fuzzy logic systems," in *Fuzzy Systems, 2006 IEEE International Conference on*. IEEE, pp. 649–654.

[144] R. K. Gupta, U. Pareek, and I. N. Kar, "Soft computation of turbine inlet temperature of gas turbine power plant using type-2 fuzzy logic systems," in *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*. IEEE, 2007, pp. 1–6.

[145] M. Ozek and Z. Akpolat, "A software tool: type-2 fuzzy logic toolbox," *Computer Applications in Engineering Education*, vol. 16, no. 2, pp. 137 – 146, 2008.

[146] H. Yazdanjouei, H. Feizy, A. Khoei, and K. Hadidi, "Design of a fully programmable analog interval type-2 triangular or trapezoidal fuzzifier," in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2012 Proceedings of the 19th International Conference*. IEEE, 2012, pp. 243–248.

[147] M. Bryk and A. Wielgus, "Digital implementation of a programmable type-2 fuzzy logic controller," *Elektronika: konstrukcje, technologie, zastosowania*, vol. 51, no. 11, pp. 44–48, 2010.

[148] A. Mesri, A. Khoei, and K. Hadidi, "Design of a current-mode fully programmable interval type-2 fuzzifier for general-purpose applications," in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2013, pp. 1–5.

[149] K. Abdulla and M. Azeem, "A novel programmable CMOS fuzzifiers using voltage-to-current converter circuit," *Advances in Fuzzy Systems*, pp. 1–9, 2012.

[150] J. Mendel, "On KM algorithms for solving type-2 fuzzy set problems," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 3, pp. 426 – 446, 2013.

[151] H. Wu and J. Mendel, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 5, pp. 622 – 639, 2002.

[152] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," in *Learning systems and intelligent robots*. Springer, 1974, pp. 1–10.

[153] P. Melin and O. Castillo, *Type 2 Fuzzy Logic*. Springer-Verlag Berlin Heidelberg, 2005.

[154] Q. Liang and J. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535 – 550, 2000.

[155] E. Antelo, T. Lang, P. Montuschi, and A. Nannarelli, "Digit-recurrence dividers with reduced logical depth," *IEEE Transactions on Computers*, vol. 54, no. 7, pp. 837–851, 2005.

[156] T. Lang and A. Nannarelli, "A radix-10 digit-recurrence division unit: algorithm and architecture," *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 727–739, 2007.

[157] N. Boullis and A. Tisserand, "On digit-recurrence division algorithms for self-timed circuits," in *International Symposium on Optical Science and Technology*. International Society for Optics and Photonics, 2001, pp. 115–125.

[158] S. Oberman and M. J. Flynn, *Fast IEEE rounding for division by functional iteration*. Computer Systems Laboratory, Stanford University, 1996.

[159] M. J. Flynn, "On division by functional iteration," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 702–706, 1970.

[160] J.-S. Chiang, E. Lai, and J.-Y. Liao, "A radix-2 non-restoring 32-b/32-b ring divider with asynchronous control scheme," *Tamkang Journal of Science and Engineering*, vol. 2, no. 1, pp. 37 – 43, 1999.

[161] S. F. Obermann and M. J. Flynn, "Division algorithms and implementations," *IEEE Transactions on Computers*, vol. 46, no. 8, pp. 833–854, 1997.

[162] A. Amaricai and O. Boncalo, "Implementation of very high radix division in FPGAs," *Electronics letters*, vol. 48, no. 18, pp. 1107–1109, 2012.

[163] P. Montuschi, L. Ciminiera, and A. Giustina, "Division unit with newton-raphson approximation and digit-by-digit refinement of the quotient," *IEE Proceedings-Computers and Digital Techniques*, vol. 141, no. 6, pp. 317–324, 1994.

[164] T. Pham, Y. Wang, and R. Li, "A variable-latency floating-point division in association with predicted quotient and fixed remainder," in *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2013, pp. 1240–1245.

[165] D. Piso and J. D. Bruguera, "A new rounding algorithm for variable latency division and square root implementations," in *Digital System Design Architectures, Methods and Tools, 2008. DSD'08. 11th EUROMICRO Conference on*. IEEE, 2008, pp. 760–767.

[166] G. Amdahl and M. Clements, "Method and apparatus for division employing table-lookup and functional iteration," Aug. 1974, US Patent 3,828,175.

[167] H. Hassler and N. Takagi, *Function evaluation by table look-up and addition*. Department of Information Science, Faculty of Engineering, Kyoto University, 1995.

[168] D. Wu and J. M. Mendel, "Aggregation using the linguistic weighted average and interval type-2 fuzzy sets," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 6, pp. 1145–1161, 2007.

[169] G. Louverdis and I. Andreadis, "Design and implementation of a fuzzy hardware structure for morphological color image processing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 3, pp. 277 – 288, 2003.

[170] M. A. Melgarejo R and C. A. Pena-Reyes, "Hardware architecture and FPGA implementation of a type-2 fuzzy system," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*. ACM, 2004, pp. 458–461.

[171] M. D. Schrieber and M. Biglarbegian, "Hardware implementation of a novel inference engine for interval type-2 fuzzy control on FPGA," Beijing, China, 2014, pp. 640 – 646.

[172] S. Guillaume, "Designing fuzzy inference systems from data: An interpretability-oriented review," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 426–443, 2001.

[173] A. Orriols-Puig, F. Martinez-Lopez, J. Casillas, and N. Lee, "A soft-computing-based method for the automatic discovery of fuzzy rules in databases: Uses for academic research and management support in marketing," *Journal of Business Research*, vol. 66, no. 9, pp. 1332 – 1337, 2013.

[174] M. Sakai, N. Homma, M. Gupta, and K. Abe, "Statistical approximation learning of discontinuous functions using simultaneous recurrent neural networks," 2002, pp. 434 – 439.

[175] S. Khrais, T. Al-Hawari, and O. Al-Araidah, "A fuzzy logic application for selecting layered manufacturing techniques," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 286 – 10 291, 2011.

[176] H. Malik, T. Mahto, B. Kr, M. Kr, and R. Jarial, "Fuzzy-logic applications in transformer diagnosis using individual and total dissolved key gas concentrations," *Journal of Electrical Engineering*, vol. 12, no. 3, pp. 202 – 206, 2012.

[177] M. Nair, R. Lakshmanan, M. Wilscy, and R. Tatavarti, "Satellite image processing for oceanic applications using fuzzy logic," *International Journal of Intelligent Systems Technologies and Applications*, vol. 10, no. 3, pp. 289 – 301, 2011.

[178] J. Ropero, C. Leon, A. Carrasco, A. Gomez, and O. Rivera, "Fuzzy logic applications for knowledge discovery: A survey," *International Journal of Advancements in Computing Technology*, vol. 3, no. 6, pp. 187 – 198, 2011.

[179] H. Nomura, "Self tuning method fuzzy reasoning by genetic algorithm," *Proc. of the Int'l Fuzzy systems and intellignet control*, vol. 4, no. 5, pp. 251–256, 1992.

[180] P. Y. Glorennec, "Adaptive fuzzy control," in *Fuzzy logic*. Springer, 1993, pp. 541–551.

[181] F. Guely and P. Siarry, "Gradient descent method for optimizing various fuzzy rule bases," in *Fuzzy Systems, 1993., Second IEEE International Conference on*. IEEE, 1993, pp. 1241–1246.

[182] O. Cordon and F. Herrera, "A two-stage evolutionary process for designing TS fuzzy rule-based systems," *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, vol. 29, no. 6, pp. 703–715, 1999.

[183] S. Sathiyan, S. Kumar, and A. Selvakumar, "Segmented fuzzy logic controller for vehicle following with optimised rule base," *Journal of Theoretical and Applied Information Technology*, vol. 57, no. 1, pp. 7 – 15, 2013.

[184] R. Arivalahan, P. Subbaraj, and D. Devaraj, "Development of genetic algorithm-based fuzzy logic controller for conical tank process," *International Journal of Industrial and Systems Engineering*, vol. 13, no. 4, pp. 442 – 461, 2013.

[185] M. Khan, M. Choudhry, M. Zeeshan, and A. Ali, "Adaptive fuzzy multivariable controller design based on genetic algorithm for an air handling unit," *Energy*, vol. 81, pp. 477 – 488, 2015.

[186] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Agent-based evolutionary approach for interpretable rule-based knowledge extraction," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 143 – 155, 2005.

[187] Y. Fu, H. Li, and M. Kaye, "Hardware/software codesign for a fuzzy autonomous road-following system," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 690 – 696, 2010.

[188] B. R. Jammu, S. K. Patra, and K. K. Mahapatra, "VLSI architecture of reduced rule base inference for run-time configurable fuzzy logic controllers," in *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*. Springer, 2013, pp. 77–88.

[189] S. Sanchez-Solano, A. Cabrera, I. Baturone, F. Moreno-Velo, and M. Brox, "FPGA implementation of embedded fuzzy controllers for robotic applications," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1937 – 1945, 2007.

[190] D. F. Chekired, C. Larbes and F. Haddad, "Implementation of a MPPT fuzzy controller for photovoltaic systems on FPGA circuit," *EnergyProcedia*, vol. 2, no. 6, pp. 541–549, 2011.

[191] A. M. A. Messai, A. Mellit and A. Guessoum, "FPGA-based implementation of a fuzzy controller MPPTfor photovoltaic module," *Energy Conversation and Management*, vol. 52, no. 1, pp. 2695–2704, 2011.

[192] M.-S. Xiao, Z. Xiao, Z.-Q. Wen, and H.-J. Yu, "Interval type fuzzifier parameter model in fuzzy c-means clustering," *Systems Engineering and Electronics*, vol. 37, no. 4, pp. 868 – 873, 2015.

[193] X. Yongjun, L. Weiguo, and O. Pengjie, "New optimized fuzzy c-means clustering algorithm," *Computer Engineering and Applications*, vol. 51, no. 11, pp. 124 – 128, 2015.

[194] W. Lu, L. Zhang, X. Liu, J. Yang, and W. Pedrycz, "A human-computer cooperation fuzzy c-means clustering with interval-valued weights," *International Journal of Intelligent Systems*, vol. 30, no. 2, pp. 81 – 98, 2015.

[195] M.-S. Xiao, Z. Xiao, Z.-Q. Wen, and H.-J. Yu, "Interval type fuzzifier parameter model in fuzzy c-means clustering," *Xi Tong Gong Cheng Yu Dian Zi Ji Shu/Systems Engineering and Electronics*, vol. 37, no. 4, pp. 868 – 873, 2015.

[196] M. Luz Lopez Garcia, R. Garcia-Rodenas, and A. Gonzalez Gomez, "K-means algorithms for functional data," *Neurocomputing*, vol. 151, no. P1, pp. 231 – 245, 2015.

[197] L. Bottou, Y. Bengio *et al.*, "Convergence properties of the k-means algorithms," *Advances in neural information processing systems*, pp. 585–592, 1995.

[198] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic contoller," *International journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[199] I. Milln, O. Montiel, R. Seplveda, and O. Castillo, *Soft Computing for Hybrid Intelligent Systems*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008, vol. 154.

[200] Y. Fu, H. Li, and M. E. Kaye, "Hardware/software codesign for a fuzzy autonomous road-following system," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 6, pp. 690–696, 2010.

Dissemination

Journal Articles

1. B R Jammu, S. K. Patra, K. K. Mahapatra, " FPGA implementation of rule optimization for stand-alone tunable fuzzy logic controller using GA,"Complex & Intelligent Systems, vol. 2, no. 2, pp. 83-98, 2016.

Journal Sumbmitted

2. B R Jammu, S. K. Patra, K. K. Mahapatra, " Tunable Type 2 Fuzzy Logic Controller with Successive Approximation based Type 2 Memebership Function"to the journal of Applied Soft Computing (Elsevier), 30 Sep 2016.

Patents Submitted

3. B. R Jammu, S. K. Patra, and K. K. Mahapatra, " Configurable inference IP core for standalone tunable fuzzy logic controller on FPGA"Submitted to NIT Rourkela Patent Cell.

Conference Presentations

4. B. R Jammu, S. K. Patra, and K. K. Mahapatra, " VLSI architecture of reduced rule base inference for run-time configurable fuzzy logic controllers,"in Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Springer, Gwalior, 2012, pp. 88-97.

5. P. Maji, B. R. Jammu, S. K. Patra, and K. K. Mahapatra, " Design and implementation of online fuzzy logic controller on FPGA,"2014 Annual IEEE India Conference (INDICON), Pune, 2014, pp. 1-5.

Author's Biodata

Name of the Candidate	:	Bhaskara Rao Jammu
Father's Name	:	Mahalakshmu Naidu Jammu
Date Of Birth	:	03 – 06 – 1982
Present Address	:	Adv. Communication Lab. Dept. of Electronics & Communication Engg. National Institute of Technology Rourkela-769 008 Odisha (India)
Permanent Address	:	H.No 5/47, Garbham Road Garividi PIN - 535101 Andhra Pradesh (India)

Academic Qualifications :

(i) B.Tech. in Electronics & Communications Engg., Andhra University, Visakhapatanm, Andhra Pradesh

(ii) M.Tech. in Electronics & Communication Engg., Motilala Nehru National Institute of Technology, Allahabd, Uttar Pradesh

Publications :

(i) Published 01 paper in International Journal

(ii) Communicated 01 paper in International Journal

(iii) Submitted 01 patent in NITR IPR Cell

(iv) Published 02 papers in International Conferences